VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

NÁVRH DÍLČÍ ČÁSTI INFORMAČNÍHO SYSTÉMU
PROPOSAL OF PART OF INFORMATION SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Petr Čmok**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jan Luhan, Ph.D., MSc**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání diplomové práce

Ústav:	Ústav informatiky
Student:	**Bc. Petr Čmok**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Informační management
Vedoucí práce:	**Ing. Jan Luhan, Ph.D., MSc**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává diplomovou práci s názvem:

Návrh dílčí části informačního systému

Charakteristika problematiky úkolu:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Vlastní návrhy řešení
Závěr
Seznam použité literatury
Přílohy

Cíle, kterých má být dosaženo:

Cílem práce je výběr metod a vytvoření zdroje dat v databázi, který bude sloužit jako podklad pro reportovací nástroj vybrané společnosti s důrazem na automatizační procesy.

Základní literární prameny:

BRUCKNER, T., J. VOŘÍŠEK, A. BUCHALCEVOVÁ a kol. Tvorba informačních systémů: Principy, metodiky, architektury. 1. vyd. Praha: Grada Publishing, 2012. 360 s. ISBN 978-80-247-4153-6.

CUESTA, H. a J. HUF. Analýza dat v praxi. 1. vyd. Brno: Computer Press, 2015, 296 s. ISBN 978-8-251-4361-2.

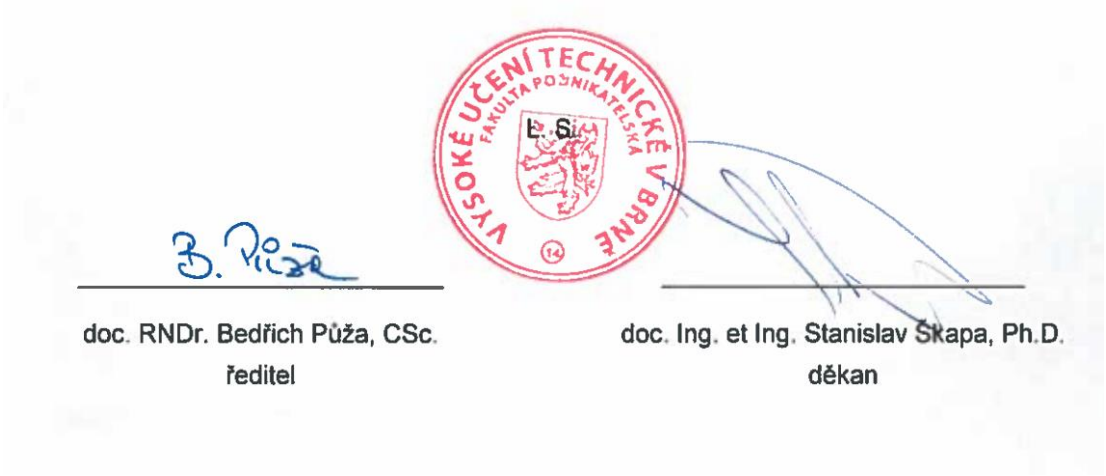
HENDL, J. Přehled statistických metod: analýza a metaanalýza dat. 4. rozš. vyd. Praha: Portál, 2012. 734 s. ISBN 978-80-262-0200-4.

KROENKE, D., D. J. AUER a J. GONER. Databáze. 1. vyd. Brno: Computer Press, 2015. 496 s. ISBN 978-80-251-4352-0.

SODOMKA, P. a H. KLČOVÁ. Informační systémy v podnikové praxi. 2. aktualiz. a rozš. vyd. Brno: Computer Press, 2010. 501 s. ISBN 978-80-251-2878-7.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2016/17

V Brně dne 28.2.2017



doc. RNDr. Bedřich Půža, CSc.
ředitel

doc. Ing. et Ing. Stanislav Škapa, Ph.D.
děkan

Abstrakt

Hlavním předmětem diplomové práce je návrh nové sekce informačního systému, který reportuje výsledky práce automatů. Cílem práce je výběr jevů a příležitostí, na které je důležité uživatele systému upozornit, nalezení vhodných metodik pro jejich automatickou detekci a příprava zadání pro frontend programátory.

Abstract

The main goal of this work is to prepare proposal for new subsegment of information system, that reports on activity of automas. The main focus is on selection events, that have meaning for users, propose the methodology for automatic detection of selected events and prepare implementation instuctions for frontend pragrammers.

Klíčová slova

Informační systém, highlights, databáze, sql, automatizace

Key words

Information system, highlights, database, sql, automatization

Bibliografická citace

ČMOK, P. *Návrh dílčí části informačního systému.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2017. 82 s. Vedoucí diplomové práce Ing. Jan Luhan, Ph.D., MSc.

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně.

Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb. O právu autorském a právech souvisejících s právem autorským).

V Brně 26. května 2017

……

Petr Čmok

Poděkování

Tímto bych rád poděkoval vedoucímu mé diplomové práce – panu Ing. Janu Luhanovi, Ph.D., MIS, za jeho odborné rady a cenné připomínky, které mi poskytoval během vytváření práce.

OBSAH

ÚVOD 11

CÍLE PRÁCE A METODIKA POSTUPU 12

1 TEORETICKÁ ČÁST PRÁCE 13

1.1 Informační systém 13

1.1.1 Data a informace 13

1.1.2 Definice informačního systému 13

1.1.3 Motivace vzniku informačního systému 14

1.1.4 Technologické pojetí informačního systému 14

1.1.5 Architektura podnikových informačních systémů 15

1.1.6 Klasifikace podnikových informačních systémů 16

1.1.7 Životní cyklus informačního systému 17

1.2 Automatizace 18

1.2.1 Automatizace jako nevyšší stupeň ve vývoji techniky 18

1.2.2 Důvody automatizace 18

1.3 Databázové systémy 19

1.3.1 Pojem databázový systém 19

1.3.2 Klasifikace databázových systémů 19

1.3.3 Databázové modely 20

1.3.4 Relační databáze 21

1.3.5 Datové sklady 24

1.3.6 Jazyk SQL 27

1.4 Vybrané části ze statistiky 29

1.4.1 Míry centrální tendence 29

1.4.2 Regulační diagramy 30

2 ANALYTICKÁ ČÁST 31

2.1 Charakter společnosti XY 31

2.2 Automatizace v oblasti správy serverů 31
2.2.1 Správa serverů z procesního pohledu 31
2.2.2 Automatizační proces 33
2.3 Reportovací aplikace 37
2.3.1 Datové struktury – zjednodušený model 37
2.3.2 Analýza použitých technologií 41
2.3.3 Uživatelská analýza 41
2.4 Řešitelné slabé stránky systému 46
3 NÁVRH VLASTNÍHO ŘEŠENÍ 47
3.1 Koncept sekce „Highlights“ 47
3.2 Analýza odchylek 47
3.2.1 Příklad využití: 47
3.2.2 Princip detekování odchylek 48
3.2.3 Příprava dat 49
3.2.4 Návrh implementace ve webovém rozhraní 52
3.3 Řešení eskalovaných ticketů 56
3.3.1 Význam 56
3.3.2 Princip řešení eskalovaných ticketů 56
3.3.3 Příprava dat 56
3.3.4 Návrh implementace ve webovém rozhraní 61
3.4 Problémy v nastavení serverů v CMDB katalogu 62
3.4.1 Význam a princip 62
3.4.2 Příprava dat 63
3.4.3 Návrh interpretace ve webovém rozhraní 65
3.5 Prezentace finálního řešení 66
3.5.1 Odchylky 66
3.5.2 Příležitosti na eskalované tickety 67

3.5.3 Problémy v CMDB katalogu 68
3.5.4 Nová verze dashboardu 68
3.6 Ekonomické zhodnocení 69
3.6.1 Přínosy mé práce 69
3.6.2 Náklady práce 70
ZÁVĚR 71
SEZNAM POUŽITÝCH ZDROJŮ 72
SEZNAM TABULEK A OBRÁZKŮ 75
SEZNAM PŘÍLOH 77

ÚVOD

Automatizace je směr, kterým se nyní ubírá téměř každá velká společnost nejen s technickým zaměřením. Mezi její hlavní přednosti patří šetření nákladů na pracovní sílu a snižování možnosti lidských chyb v procesech, kde je nasezena. To má positivní vliv jak na ceny, tak na kvalitu výsledného produktu či služby. Automatizace je nekončící proces, který je potřeba neustále zlepšovat.

Klíčovou roli při zlepšování procesu mají informace o jeho aktuálním stavu. Reportovací nástroj, který je v této diplomové práci popsán, se zabývá sběrem dat z automatizovaného prostředí a prezentuje je uživateli v podobě přehledných informací. Cílem této práce je návrh komplexního systému upozornění, který bude uživatele navádět, jak zlepšit výsledky automatizačního procesu.

Práce je strukturována do tří částí – teoretický úvod, analytická část a návrh vlastního řešení problému.

V teoretickém úvodu bude čtenář seznámen s pojmem informační systém, motivací pro vznik informačního systému, klasifikací podnikových informačních systémů a jeho životním cyklem. Ve druhé části teoretického úvodu je obecně popsána automatizace, jako nejvyšší stupeň ve vývoji techniky a jsou zde představeny hlavní důvody pro zavádění automatizace. Třetí kapitola se zabývá představením databázových systémů, s důrazem na jejich klasifikaci, relační databáze a datové sklady. Poslední kapitola teoretické části obsahuje základní přehled statistických metod, které byly použity při hledání návrhu řešení této práce.

Analytická část rámcově představí zadavatelskou společnost XY. Společnost si z důvodu citlivosti dat a konkurence v prostředí nepřála být v této práci uvedena. Následuje zjednodušené představení správy serverů ve společnosti a vlivu zavedení automatizace. Další část uživatele seznámí se samotným automatizačním procesem a entitami, které se v něm vyskytují. Ve třetí části je představena samotná reportovací aplikace s důrazem na databázové struktury a uživatelské prostředí. Analýza je zakončena shrnutím řešitelných slabých stránek systému.

Návrh řešení spočívá ve vytvoření zadání front-end programátorům, které bude obsahovat návrhy na řešení vybraných slabých stránek, které byly odhaleny v analytické části.

CÍLE PRÁCE A METODIKA POSTUPU

Cílem práce je výběr metod a vytvoření zdroje dat v databázi, který bude sloužit jako podklad pro reportovací nástroj vybrané společnosti s důrazem na automatizační procesy.

Dílčí cíle:

- vytvoření teoretických základů z oblasti informačních systémů, automatizace a databázových systémů, které čtenáři usnadní pochopení této práce,
- analýza současného stavu z uživatelského pohledu a odhalení významných jevů, jejichž vyřešení může sloužit ke zlepšení výsledků automatizace ve společnosti,
- vytvoření návrhů na automatické identifikování těchto jevů,
- vytvoření návrhu na zapracování systému upozornění do nové verze reportovací aplikace,
- výstupy této práce budou vývojové diagramy a připravené dotazy v jazyku SQL, které budou sloužit jako zadání pro front-end programátory.

.

1 TEORETICKÁ ČÁST PRÁCE

Teoretický úvod práce je rozdělen na čtyři hlavní kategorie – informační systémy, automatizace, databázové systémy a vybrané části ze statistiky.

1.1 Informační systém

V této části práce bude informační systém definován z různých úhlů pohledu, budou představeny jeho různé klasifikace a jeho životní cyklus.

1.1.1 Data a informace

Data jsou ve světě informačních technologií chápány jako čistá fakta. Ty mohou mít různé formy, ať už se jedná o číslo zapsané na papír, nebo větu převedenou na bity a byty, které jsou uloženy v paměti počítače. Zabezpečení, zaznamenávání a využívání těchto dat je pro úspěšnou společnost nezbytným procesem. [1]

S definicí dat úzce souvisí definice **informací**. Informace jsou chápány jako data, která byly zpracovány tak, že mají pro uživatele význam. Proces získávání informací se skládá ze sběru dat, které se následně podrobeny transofrmaci, za účelem vytvoření informací. Příkladem dat může být záznam o tržbách, za informace mohou být v tomto případě považovány např. přehledy o nejprodávanějších produktech v daných lokalitách. [1]

Při práci s informačními systémy nesmí být opomíjena stránka použitelnosti prezentovaných informací. Při rozlišení „dobrých“ a „špatných“ informací, se musí brát v úvahu vlastnosti spojené s časem, obsahem a formou, kterou nám byla informace podána. [1]

1.1.2 Definice informačního systému

Na termín informační systém existuje mnoho různých pohledů. Jedna z možných definic informačního systému ho popisuje, jako soubor lidí, technologických prostředků a metod, které spolu zajišťují sběr, přenos, zpracování a uchování dat za účelem prezentování informací pro potřeby uživatelů. Příkladem informačního systému může být telefonní seznam, kartotéka nebo účetnictví. [3]

Takto definovaný informační systém, nemusí být nutně automatizovaný pomocí počítačů, ale může se vyskytovat např. v papírové podobě. S moderním vnímání informační

systému dnes úzce souvisí pojem informační a komunikační technologie. Pod tento pojem je možné zařadit všechny komunikační zařízení a aplikace, které na nich fungují. [4]

1.1.3 Motivace vzniku informačního systému

Informační systémy hrají podstatnou roli v provozu hospodářských subjektů a mohou výrazně zlepšit jejich konkurenceschopnost. Mezi hlavní přínosy informačních systémů patří:

- rychlé získávání, zpracování a předání informace,
- efektivnější způsoby komunikace se zákazníky a partnery,
- personalizace produkce v hromadné výrobě,
- podpora samoobslužných procesů,
- kvalitnější způsoby uchování dat pomocí jejich digitalizace,
- prodeje po internetu,
- tyto služby poskytují 24 hodin denně, 365 dní v roce. [5]

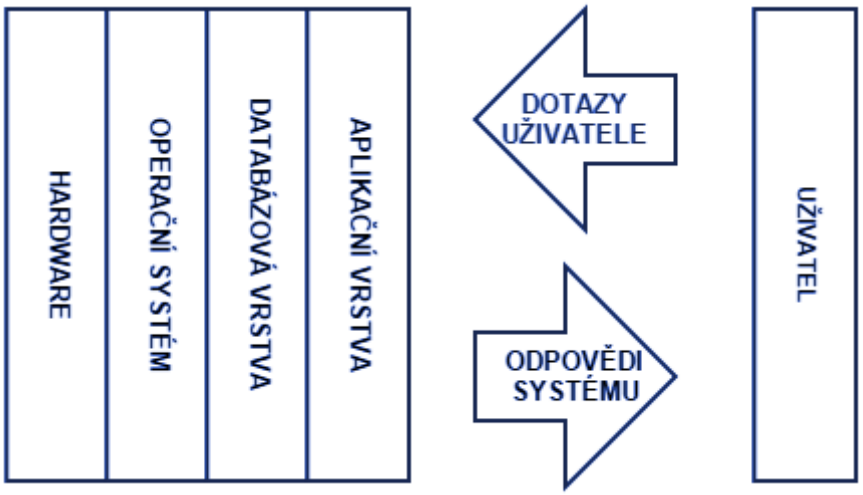
Informatizace společnosti zvyšuje kvalifikaci obyvatel v IS/ICT a tak zvyšuje výkonnost ekonomiky a generuje nové trhy. Byznys manažer dnes musí umět využívat ICT pro efektní řízení firmy a jejich procesů. S ohledem na roli ICT v současné ekonomice porostou i v tomto desetiletí investice podniků do ICT. [5]

1.1.4 Technologické pojetí informačního systému

Tradiční informační systém je možné z technologického pohledu vnímat z pěti vrstev:

- hardware,
- operační systém
- databázová platforma,
- aplikační software
- uživatelé.

Uživatel komunikuje s celým systémem prostřednictvím aplikačního softwaru (aplikační vrstvy). Komunikace mezi dalšími vrstvami je plně automatická. [6]



Obrázek 1: Vrstvy informačního systému, z [6]

1.1.5 Architektura podnikových informačních systémů

Architekturu podnikového informačního systému je možné rozdělit do následujících skupin:

- dvouvrstvá s výkonem soustředným u klienta
- dvouvrstvá s výkonem soustředným na serveru
- třívrstvá architektura
- n-vrstvá architektura.

Dvouvrstvá architektura s výkonem soustředěným u klienta funguje tak, že veškeré aplikační a uživatelské služby se zpracovávají na klientské stanici uživatele. Tato architektura často bývá označováno jako tlustý klient. Mezi klientem a serverem probíhá velký počet datových přenosů. [6]

Dvouvrstvá architektura s výkonem soustředěným na serveru se stává v posledních letech velice populární řešení. Princip této architektury spočívá v tom, že se na straně klienta zobrazuje pouze uživatelské prostředí a většina výpočetních operací se děje na serveru. Toto uspořádání bývá také označováno jako tenký klient. Hlavní výhody tohoto řešení jsou snadná škálovatelnost celého systému a nižší nároky na síťovou infrastrukturu. Mezi nevýhody je možné zmínit vysoké nároky výpočetní sílu serveru. [6]

Třívrstvá architektura se od dvouvrstvé liší tak, že jsou od sebe databázové a aplikační služby odděleny do samostatných logických celků. Klient opět pracuje pouze

s uživatelským rozhraním. Třívrstvý model vykazuje vyšší stabilitu, neboť je zátěž provozu celého řešení rozložena na více serverů. [6]

N-vrstvá architektura rozděluje aplikaci na n serverů do n logických celků. Toto řešení je vhodné u velice komplexních aplikací. [6]

Při představování jednotlivých druhů architektur je nutné představit i pojem **virtualizace.** Virtualizace umožňuje k dostupným zdrojům přistupovat jiným způsobem, než jsou fyzicky uspořádány. Mezi hlavní přínosy virtualizace patří snižování nákladů při pořizování hardwaru, úspory energie, zvýšení efektivity správy IS/ICT a snadné přizpůsobování architektury uživatelským potřebám. [6]

1.1.6 Klasifikace podnikových informačních systémů

V následující části budou představeny základní rozdělení podnikových informačních systémů.

1.1.6.1 Funkcionální přístup

Jedná se o tradiční rozdělení aplikací v informačním systému, kde každý software (nebo modul softwaru) zastává určitou funkci. Informační systém se může například skládat z účetního softwaru, softwaru na evidenci zásob, personálního softwaru a dalších softwarů, které zastávají jednu, jasně definovanou funkci. [6]

1.1.6.2 Holisticko-procesní přístup

Podnikové informační systéme je možné klasifikovat dle jejich praktického uplatnění při základních podnikových procesech. Tento přístup se v praxi označuje jako holisticko-procesní pohled. Podnikové informační systém se dle tohoto podniku dělí následovně: [6]

- ERP je chápáno jako jádro informačního systému, zaměřené na řízení interních podnikových procesů,
- CRM systém obsluhuje procesy směřované k zákazníkům,
- SCM obsahuje procesy směřované k dodavatelům,
- MIS systém pro podporu manažerského rozhodování. Sbírá data z ERP, CRM a SCM systémů a na jejich základě poskytuje informace pro podporu rozhodovacího procesu podnikovému managementu. [6], [7]

1.1.7 Životní cyklus informačního systému

Životní cyklus informačního systému je možné chápat jako za sebou jdoucí období. Každé toto období má jasně definovaný cíl, na základě kterého jsou koordinovány všechny činnosti související s daným infomačním systémem. Etapy životního cyklu informačního systému:

- předběžná analýza, neboli specifikace cílů,
- analýza systému, neboli specifikace požadavků,
- projektová studie, neboli návrh,
- implementace,
- testování,
- zavádění systému,
- zkušební provoz,
- rutinní provoz a údržba,
- reengineering. [8]

V praxi se setkáme s různými podobami tohoto rozdělení, kde se jednotlivé etapy přizpůsobují danému projektu. [8]

1.2 Automatizace

Pojmem automatizace se označuje použití řídicích systémů (např. počítačů) k řízení průmyslových procesů a zařízení. Zatímco mechanizace poskytuje lidem k práci zařízení, které jim práci usnadňuje, automatizace snižuje potřebu přítomnosti člověka při vykonávání určité činnosti. Automatem je možné označit zařízení, které samostatně vykonává předem stanovené úkony. [9]

1.2.1 Automatizace jako nevyšší stupeň ve vývoji techniky

Automatizace představuje nevyšší stupeň ve vývoji techniky:

- stupeň - instrumentace (fyzická práce za pomocí ručních nástrojů),
- stupeň - mechanizace (fyzická práce nahrazována stroji),
- stupeň – automatizace (fyzická i duševní práce nahrazována činností strojů). [10]

1.2.2 Důvody automatizace

Hlavní důvody pro automatizace je možné rozdělit do dvou kategorií:

- **Vynucená automatizace –** náhrada lidské práce je vynucena okolnostmi:
    - činnost lidského pracovníka je příčinou chyb a vede ke ztrátám,
    - jeho přímá účast mu způsobuje zdravotní potíže,
    - pracovník není schopen vykonat činnost z hlediska rychlosti, přesnosti, rozsahu nebo jiných příčin,
    - není možná přítomnost lidské obsluhy,
    - nedostatek kvalifikovaných lidských pracovníků.
- **Ekonomické důvody** pro automatizaci:
    - zavedení automatizace představuje snížení nákladů, zvýšení jakosti, zvýšení produktivity, či jiné konkurenční výhodě. [9]

1.3 Databázové systémy

Příprava datových podkladů bude tvořit výraznou část návrhu řešení této práce. V této části práce budou představeny základní pojmy spojené s tímto tématem, se důrazem na klasifikaci databázových systémů, relační databáze a datové sklady.

1.3.1 Pojem databázový systém

Databázové systémy se skládají z databáze a systému řízení báze dat.

Databáze je chápána jako uspořádaná sdílená množina dat a metadat, která je uložena na datovém médiu. Pojem data byl zmíněn v kapitole o informačních systémech, metadata popisují základní charakteristiky a principy uvnitř databáze. [11]

Systém řízení báze dat (dále jen SŘBD) je soubor programů, které spravují databázovou strukturu a dovolují nám získávat potřebné informace v organizované formě. Používání SŘBD přináší oproti přímému napojení databáze na aplikaci řadu výhod:

- díky centralizovaným funkcím a procedurám na databázovém serveru, umožňuje vytvořit **lepší a stabilnější prostředí pro sdílení databází přes více aplikací**,
- rychlejší programování koncových aplikací,
- SŘBD umožňuje kontrolovat **integritu dat**, centralizovaně přímo v databázi a tak zabraňuje nedostatečným kontrolám, které by se mohli objevovat u jednotlivých aplikací, které k databázi přistupují,
- **větší produktivita systému** díky rozdělení zátěže na hardware mezi uživatelskou stanici a server. [11]

1.3.2 Klasifikace databázových systémů

Databázové systémy je možné rozdělit na základě tří faktorů:

- dle počtu uživatelů,
- dle lokace,
- dle využití. [11]

1.3.2.1 Dle počtu uživatelů

Single-user databáze podporují pouze jednoho uživatele v daný okamžik. V praxi to znamená, že pokud uživatel A operuje s databází, uživatel B a C musí čekat,

než uživatel A dokončí svou činnost. Oproti tomu **multiuser databáze** podporuje více uživatelů vdaný okamžik. [11]

1.3.2.2 Dle lokace

Centralizovanou databázi je možné označit databázi, která má všechna úložiště na jednom místě. Mezi hlavní výhody centralizované databáze patří snadnější přístup k datům a analytickým službám. [11]

Distribuované databáze jsou naproti tomu umístněné na více místech. Lokace těchto dat je zvolena podle toho, kde jsou tyto data nejčastěji využívány. Výhodou tohoto řešení je nižší hardwarová náročnost. [11]

1.3.2.3 Dle využití

Pojmem **operační databáze** je možné označit ty databázové systémy, které jsou zaměřeny a optimalizovány na běžné operace s daty – vkládání nových záznamů, jejich úprava či mazání. [11]

Datové sklady jsou databáze s velkým množstvím dat, které jsou využívány pro získávání souhrnných informací v reálném čase. [11]

1.3.3 Databázové modely

V následující sekci budou popsány základní databázové modely.

1.3.3.1 Hierarchický model

V tomto modelu jsou data strukturována hierarchicky a obvykle se znázorňují jako obrácený strom. Základní tabulka modelu slouží jako „kořen“ obráceného stromu a ostatní tabulky z něj vychází. Vztahy v hierarchickém modelu jsou reprezentovány termíny rodič a potomek. K datům se v hierarchickém modelu přistupuje tak, že se začne v kořenové tabulce a postupně se přes stromovou strukturu propracovává k hledaným datům. Hlavní nevýhodou je nutná znalost celé struktury databáze. [12]

1.3.3.2 Síťový model

Síťový model je zobecnění a vylepšení hierarchického modelu. Struktura je v tomto případě vyjádřena pomocí uzlů (soubor záznamů) a množinových struktur (vztahy). Jeden uzel je zde definován jako vlastník a druhý jako prvek. Hlavní výhodou oproti hierarchickému modelu je, že jeden vlastník může mít více prvků. [12]

1.3.3.3 Relační model

Základem relačního modelu je matematické zobecnění pojmu soubor, pomocí silného formalizmu matematické relace a využívání operací relační algebry. Model má jednoduchou strukturu, data jsou organizována v tabulkách (relacích), které se skládají z řádků (záznamů) a sloupců (atributů). Podstatnou roli v tomto modelu hrají vztahy, mezi jednotlivými relacemi. [12]

1.3.3.4 Objektový model

Hlavní myšlenkou objektově orientovaného modelu je, že databázový vývojář se stará o všechny aspekty databáze z objektově orientovaného programu. Objektově orientovaný datový model obsahuje všechny charakteristiky objektově orientovaných jazyků. Data jsou čerpány z relačních databází, které jsou však v tomto případě degradovány na pouhé skladiště dat. Neexistuje zde jasné rozdělení mezi databázovým programem a aplikačním softwarem. [12]

1.3.3.5 Objektově relační model

Objektově relační model je relační model, rozšířený o softwarovou vrstvu, která dodává relačnímu SŘBD objektový charakter a bohatší typový systém. Mezi hlavní výhody patří možnost využití tříd, zapouzdření a dědičnosti. Cílem při vytvářené tohoto modelu bylo, aby relační databáze dokázala zpracovávat i složitější datové typy, jako například audio záznamy, video i složité stavební nákresy. [12]

1.3.4 Relační databáze

Relační databáze byla představena v roce 1969 pracovník firmy IBM - Dr. E. F. Codd a v současnosti je nejrozšířenější model používaný při správě databází. [12]

1.3.4.1 Struktura databáze

Relační databázový model sdružuje data do relací (tabulek). Sloupce této tabulky se v jazyku relačních databází nazývají atributy, řádky n-tice nebo také záznamy. Každý atribut má jasně definován jednoznačný název, typ a rozsah. Průnik daného atributu a záznamu se nazývá hodnota. Se strukturou databáze úzce souvisí pohledy. Pohledy je možné chápat jako virtuální databázové tabulky, které vznikly spojením polí z jedné nebo více tabulek. [12], [13]

1.3.4.2 Klíče

Klíče v relačních databázích zabezpečují jednoznačnou identifikace daného záznamu, napomáhají dodržení různých druhů integrity a slouží jako prostředek pro definování vztahů. V následující části budou popsány jednotlivé druhy klíč. [12]

Kandidát na klíč je jedno nebo více polí, které jednoznačně identifikují záznamy v relacích. Kandidát na klíč musí mít splňovat následující podmínky: [12]

- nesmí být vícesložkové pole,
- musí obsahovat jedinečné hodnoty,
- nesmí obsahovat hodnoty NULL,
- jeho hodnota nesmí být tajná,
- hodnota nesmí být volitelná,
- skládá se z nejmenšího možného počtu polí,
- jeho hodnota jednoznačně identifikuje hodnoty všech polí daného záznamu,
- jeho hodnota je během běžného provozu neměnná. [12]

Ze všech možných kandidátů na klíč je vybrán nejvhodnější a označen jako **primární klíč**. Kandidáti na klíč, kteří se nestanou primárním klíčem, označujeme jako **alternativní klíče.** Kandidátní klíč může být uměle vytvořen. [12]

Pole s primárním klíčem jednoznačně identifikuje záznamy v rámci tabulky i v rámci celé databázové struktury. Každá tabulka musí mít pouze jeden primární klíč. Primární klíč složený z více než jednoho pole se nazývá složený primární klíč. [12]

Atribut relace může nazvat **cizím klíčem**, pokud splňuje tyto vlastnosti:

- Každá hodnota cizího klíče je buď plně zadaná, nebo plně nezadaná.
- Existuje relace s kandidátním klíčem takovým, že hodnota cizího klíče je identická s hodnotou kandidátního klíče relace.

Soulad hodnot cizích a primárních klíčů představuje vztahy mezi řádky tabulek a tak „drží“ integritu dat databáze. Ostatní atributy relace jsou označeny jako **neklíč**. [12]

1.3.4.3 Vztahy mezi relacemi

Vztahy mezi relacemi je možné popsat pomocí těchto vlastností:

- stupeň vztahu,
- kardinalita vztahu,
- volitelnost účasti ve vztahu.

Stupeň vztahu definuje počet účastníků ve vztahu. Z tohoto pohledu je možné rozlišit následující typy vztahů:

- unární - vztah sama na sebe,
- binární - vztah mezi dvěma relacemi,
- ternární - vztah mezi třemi relacemi,
- n-ární - vztah mezi n-relacemi zároveň. [14]

Druhá vlastnost, která určuje podobu vztahu je jeho **kardinalita**. Ta může nabývat následujících hodnot:

- Mezi tabulkami není vztah.
- Vztah 1:1 – se používá, pokud jednomu záznamu odpovídá právě jeden záznam v jiné databázové tabulce. Tento typ vztahu není v praxi příliš využíván, protože většinou neexistuje pádný důvod, proč takovéto záznamy neumístit do jedné databázové tabulky.
- Vztah 1:N - přiřazuje jednomu záznamu více záznamů z jiné tabulky. Jedná se o nejpoužívanější typ relace.
- Vztah M:N - umožňuje M záznamům z jedné tabulky přiřadit N záznamů z tabulky druhé. V databázové praxi bývá tento vztah dekomponován na kombinaci dvou vztahů 1:N a 1:M.

Poslední vlastností, která definuje vztah mezi relacemi, je **volitelnost účasti** ve vztahu. Ta určuje zda je účast ve vztahu povinná, nebo volitelná. [14]

1.3.4.4 Integrita databáze

Ve světě relačních databází je možné rozlišovat následující druhy integritních omezení:

- **Doménové integritní omezení** definuje platné hodnoty pro atribut, pomocí výběru logického datového typu, definování měřítka, definování přesnosti a nakládání s neznámými hodnotami.

- **Omezení přechodové integrity** definuje stavy, kterým můžou dané hodnotě předcházet.
- **Entitní integritní omezení** zajišťuje, aby nebylo možné vložit do databáze duplicitní záznamy. Za entitní omezení lze považovat existenci správně navrhnutého primárního klíče.
- **Referenční integrita** ochraňuje vazby mezi relacemi. V praxi to znamená, že žádný řádek v cizí tabulce nesmí obsahovat takovou hodnotu cizího klíče, která nemá odpovídající záznam v primární tabulce.
- **Databázová integrita** zachycuje omezení týkající se obsahu databáze.
- **Omezení transakční integrity** ovládají způsoby přípustné manipulace s daty. Jsou procedurálního charakteru a nejsou součástí datového modelu. [14]

Vývojář má dnes k dispozici velké množství nástrojů, pomocí který může integritu databáze zajistit. Mezi ty nejpoužívanější patří spouště a opatření na straně straně uživatelského rozhraní. [15]

1.3.4.5 Normální formy

Normální formy je možné chápat jako doporučení pro správný chod relační databáze.

- Relace je v **první normální formě**, právě když všechny její domény obsahují pouze atomické hodnoty.
- Relace je ve **druhé normální formě**, právě když je v 1NF a každý její neklíčový atribut, je plně závislý na každém kandidátním klíči.
- Relace je ve **třetí normální formě**, právě když je ve 2NF a neexistuje žádný neklíčový atribut, který je tranzitivně závislý na některém kandidátním klíči.

Existuje i čtvrtá a pátá normální forma, ty jsou ovšem při praktickém návrhu zvažovány jen zřídka. Nepřihlížení k těmto pravidlům může vést k méně dokonalému návrhu, ale neovlivní funkčnost databáze. [13], [16]

1.3.5 Datové sklady

Datovým skladem je možné označit zvláštní typ relační databáze, která umožňuje řešit úlohy zaměřené na analytické dotazování nad rozsáhlými datovými soubory dat. [17]

1.3.5.1 Definování datového skladu

Datové sklady je možné chápat jako kolekci dat, která slouží k podpoře rozhodovacího procesu managementu. Jeho klíčové charakteristiky jsou: [17]

Orientace na subjekt

Oproti běžné operační databázi, jsou zde data ukládány tak, aby co nejpřesněji reprezentovaly strukturu objektů z reálného světa. Oproti operační databáze se zde často nedodržuje doporučení definované normální formami. Výsledek je pro uživatele čitelnější struktura za cenu zvýšených nároků na paměťové prostory. [17]

Integrovaná data z více aplikací

Běžná provozní aplikace nad relační databází řeší určitý specifický okruh úloh nad „svými“ specifickými daty. V datovém skladu je nutné shromáždit informace z mnoha různých zdrojů a seskupit je podle logického významu (úzce souvisí s **orientací na subjekt**). [17]

Historie dat

Operační databáze v sobě často ukládají pouze současné datové hodnoty, které jsou významné pro provoz této aplikace. Datové sklady by měly umožňovat analýzu minulosti, zobrazit informace o současnosti a poskytovat podklady pro odhadování budoucího vývoje. Ze své podstaty tedy musí obsahovat všechna dostupná historická data. [17]

Nevratné údaje

Data jsou do datového skladu nahrávána ve větších dávkách (například v denních nebo týdenních intervalech) a pak již nejsou nijak modifikována. Při ETL procesu je tedy nutné zajistit, aby nahraná data byla v konečném stavu (nenahrávat data, u kterých je možné předpokládat další změny). [17]

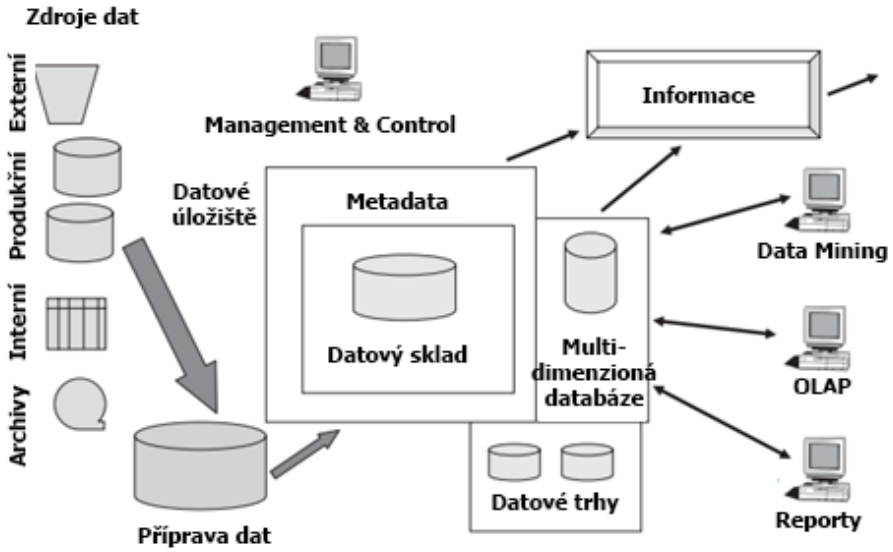
Zrnitost dat

Další významný rozdíl datového skladu oproti operační databáze je, že v datovém skladu často volíme míru detailu uložených dat dle úrovně, na které se právě nacházíme. Měsíční agregované údaje budou mít větší úroveň detailu než denní přehledy. [17]

Další charakteristiky datového skladu

Datový sklad musí obsahovat nástroj pro nahrávání dat z různých datových zdrojů. Tyto zdroje mohou mít různé datové formáty a různé fyzické umístění. Nemusí se jednat pouze o relační databáze. Zároveň musí ukládat data s ohledem na nejlepší a nejrychlejší provádění složitých dotazů. [17]

1.3.5.2 Komponenty datového skladu



Obrázek 2: Komponenty datového skladu, [17]

Zdrojová data pro tvorbu datového skladu je možné rozdělit do čtyř kategorií:

- produkční data,
- interní data,
- data z archivů,
- externí data. [17]

Tyto data jsou nahrány do komponenty pro stahování a přípravu dat, kde jsou následně transformovány do žádaného tvaru. Po této transformaci, jsou data nahrány do datového skladu, kde k nim přistupují aplikace pro prezentování informací. [17]

1.3.5.3 Struktura datového skladu

Data v datovém skladu jsou členěna do **schémat**. Každé schéma odpovídá jedné analyzované funkční oblasti. Jádro každého schématu tvoří **tabulky faktů**, ve kterých

jsou uložena analyzovaná data – hodnoty, které jsou použity k analytickým výpočtům. Faktové tabulky jsou pomocí cizích klíčů spojeny s **dimenzemi**. Dimenze jsou tabulky, které obsahují seznamy hodnot sloužících ke kategorizaci a třídění dat ve faktových tabulkách. V praxi se často setkáváme s případy, kde tabulka dimenzí není napojena přímo na tabulku faktů, ale na jinou tabulku dimenzí, vnikají tak tzv. **hierarchické dimenze**. [18]

Při ukládání hierarchických dimenzí existují dvě možnosti:

- Schéma hvězda – případ, kde je z celé hierarchie vytvořena jedna tabulka dimenzí, která se váže na tabulku faktů. Výsledné schéma připomíná tvar hvězdy.
- Schéma vločka – případ, kde je zachováno větvení celé hierarchie. Výsledné schéma připomíná tvar vločky. [18]

1.3.6 Jazyk SQL

Příkazy jazyka SQL umožňují úplnou kontrolu nad systémem řízení báze dat. Podle svého účelu je možné je rozdělit do následujících skupin:

- DML – příkazy pro manipulaci s daty,
- DDL – příkazy pro definici dat,
- DCL – příkazy pro řízení dat.[19]

Příkazy pro manipulaci s daty

Jsou to příkazy pro získání dat z databáze a pro jejich úpravy. Označují se zkráceně **DML** (zkratka z anglického Data Manipulation Language - jazyk pro manipulaci s daty**).**

- SELECT – vybírá data nebo jejich podmnožinu z databáze, umožňuje výběr podmnožiny a řazení dat,
- INSERT – vkládá nová data do databáze,
- UPDATE – mění data v databázi,
- DELETE – odstraňuje data (záznamy) z databáze,
- REPLACE – shodný s příkazem insert, pokud ale v tabulce hodnota již existuje, je nahrazena novou hodnotou,
- TRUNCATE – odstraní všechna data v tabulce. [19]

Příkazy pro definici dat

Těmito příkazy se vytvářejí struktury databáze – tabulky, indexy, pohledy a další objekty. Označují se zkráceně **DDL** (zkratka z anglického Data Definition Language - jazyk pro definici dat**)**.

- CREATE – vytváření nových objektů,
- ALTER – změny existujících objektů,
- DROP – odstraňování objektů. [19]

Příkazy pro řízení dat

Do této skupiny patří příkazy pro nastavování přístupových práv a řízení transakcí. Označují se zkráceně **DCL** (zkratka z anglického Data Control Language - jazyk pro ovládání transakcí**)**.

- START TRANSACTION – příkaz, který zahájí transakci,
- COMMIT – potvrzení transakce,
- ROLLBACK – zrušení transakce, návrat do původního stavu. [19]

1.4 Vybrané části ze statistiky

Statistika je naukou o získávání kvantitativních údajů (dat) a o jejich analýze. [20]

1.4.1 Míry centrální tendence

Míry centrální tendence (také označovány jako hodnoty polohy) se používají pro určení typické hodnoty dat. Nejrozšířenější z nich jsou aritmetický průměr, modus a medián. Další míry centrální tendence mají úzké spektrum využití a pro tuto práci jsou nevýznamné. [20]

1.4.1.1 Aritmetický průměr

„*Aritmetický průměr* $\bar{x}$ *hodnot x* $x_1, x_2, \ldots, x_n$ *je definován vztahem*

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n},$$

n je celkový počet pozorování.“ [21, str. 41]

1.4.1.2 Modus

„*Modus* $\hat{x}$ *je hodnota znaku s největší četností.*“ [21, str. 44]

1.4.1.3 Medián

„Kvantil x_p *je reálné číslo, pro které platí, že 100*p% jednotek uspořádaného souboru, má hodnotu menší nebo rovnu* x_p *a 100*(1-p)% jednotek má větší nebo rovnu* x_p*“* [21, str. 45]

Medián x̃ je speciální název pro kvantil $x_{0.50}$. [21]

1.4.1.4 Použití měr centrální tendence

Rozhodnutí, kterou charakteristiku průměru nebo polohy použijeme při popisu dat, závisí na cílech a předpokladech analýzy. Jestliže jsou data symetricky rozdělaná, všechny tyto charakteristiky jsou přibližně stejné. [20]

Aritmetický průměr je vhodné použít, pokud:

- jestliže data nejsou kategoriální,
- jestliže je rozdělení symetrické,
- jestliže chceme použít statistické testy. [20]

Medián je vhodné použít, pokud:

- jestliže jsou data získaná minimálně ordinálním měřítku,
- jestliže chceme znát střed rozdělení dat,
- jestliže data mohou obsahovat odlehlé hodnoty,
- jestliže je rozdělení dat silně zešikmené. [20]

Modus je vhodné použít, pokud:

- má rozdělení více vrcholů,
- jestliže chceme získat pouze základní přehled o rozdělení,
- jestliže se slovem průměr míní nejčastější hodnota. [20]

1.4.2 Regulační diagramy

Regulační diagram je grafický prostředek pro zobrazení vývoje variability procesu v čase, popisuje jeho zvládnutelnost a poskytuje signál o tom, že na proces začala působit vymezitelná příčina.

Graficky se regulační diagram sestrojuje následovně:

- Na vodorovné ose je znázorněn čas, na svislé ose jsou zobrazeny jednotlivá čísla logických podskupin, dle pořadí jejich získání.
- Na svislou osu vyneseme stupnici pro charakteristiku zkoumané regulované veličiny, která je ve zvoleném regulačním diagramu použita jako testové kritérium stability procesu.
- Do takto připraveného grafu, jsou zaneseny hodnoty charakteristiky regulované veličiny, získané v jednotlivých logických podskupinách. Pro pozorování trendu je možné tyto body spojit přímkou.
- Pro podporu rozhodnutí o statistické zvládnutelnosti procesu slouží přímky:
    - UCL – hodní regulační mez vymezuje pásmo působení pouze náhodných příčin variability procesu.
    - UWL – přímka ve dvou třetinách vzdálenosti UCL od střední přímky
    - CL – centrální přímka, která odpovídá požadované referenční hodnotě použité charakteristiky.
    - LWL – přímka ve dvou třetinách vzdálenosti LCL od střední přímky.
    - LCL - dolní regulační mez vymezuje pásmo působení pouze náhodných příčin variability procesu. [22]

2 ANALYTICKÁ ČÁST

V této části bude nejprve rámcově představena společnost, pro kterou je práce vypracována. Ve druhé části bude čtenář rámcově seznámen se základními automatizačními procesy a jednotlivými entitami, které se v tomto procesu vyskytují. Následně bude analyzován současný stav reportovacího nástroje a návrh jeho nové verze, do které budou vylepšení popsané v této práci nasazeny.

2.1 Charakter společnosti XY

Společnost XY byla založena v 20. letech devatenáctého století v USA a patří mezi největší světové hráče působící v oblasti informačních technologií. Mezi hlavní činnosti patří výroba a prodej počítačového software, hardware a poskytování IT služeb. Společnost má několik set tisíc zaměstnanců po celém světě. V Brně má v současné době zhruba 4000 zaměstnanců.

Jedna ze služeb, kterou společnost nabízí je poskytování serverů jako služby. Tým, pro který je tato práce vytvořena, se věnuje programování automatů, které řeší jednoduché opakující se incidenty na těchto serverech.

2.2 Automatizace v oblasti správy serverů

V této části bude nejprve představen proces správy serverů před automatizací a po automatizaci. Následně budou popsány entity, které se v tomto procesu vyskytují, a pomocí nich bude popsán automatizační proces.

2.2.1 Správa serverů z procesního pohledu

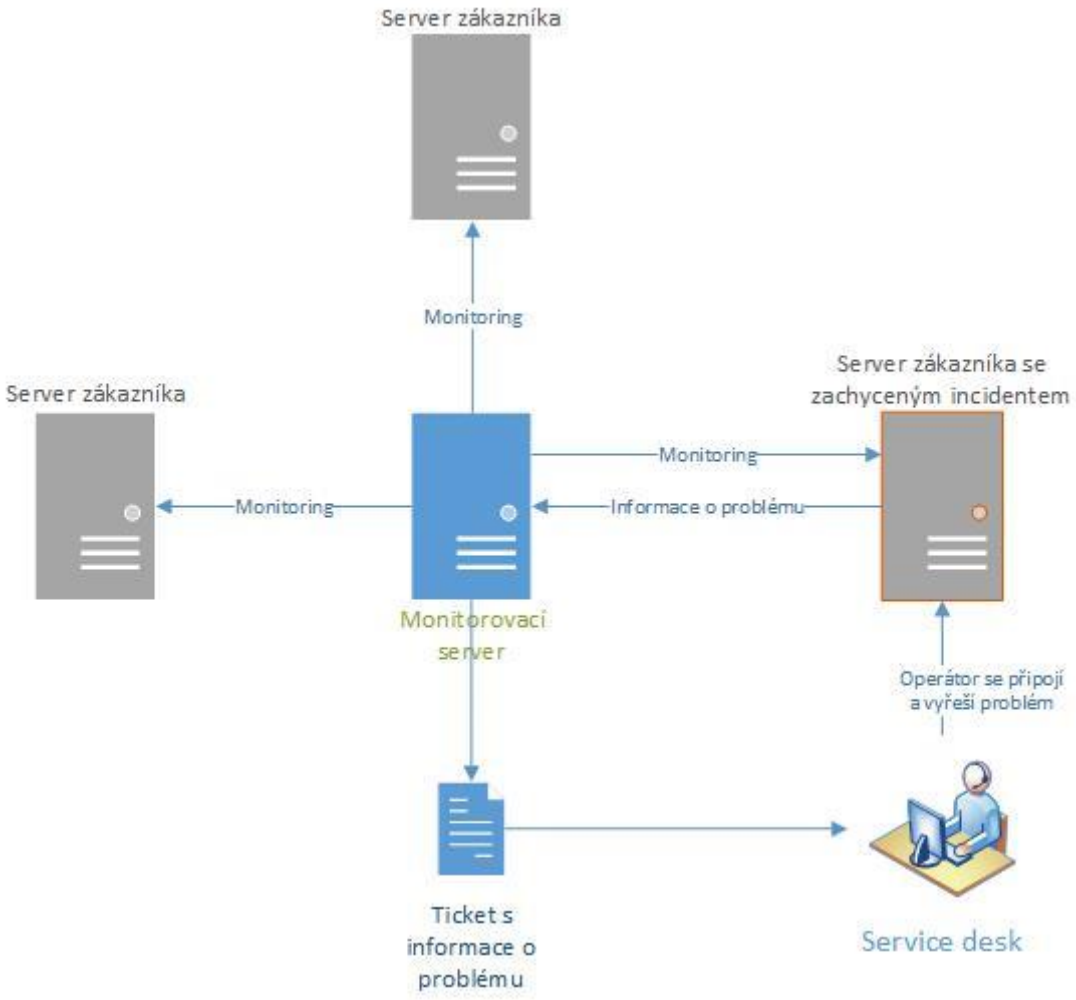
Automatizace byla v této oblasti zavedena z ekonomických důvodů, protože šetří čas kvalifikovaných pracovníků, a tak i celkové náklady pro všechny zainteresované strany. Automaty řeší pouze jednoduché opakující se problémy. Komplexní incidenty musí stále řešit technik s odpovídajícími odbornými znalostmi.

2.2.1.1 Před automatizací

Před spuštěním automatizace proces správy severů ve vybrané společnosti vypadal následovně:

Nad serverem zákazníka byla provozována monitorovací služba, která zachytávala všechny problémy, které se na těchto serverech vytvářely. Tato služba při zachycení

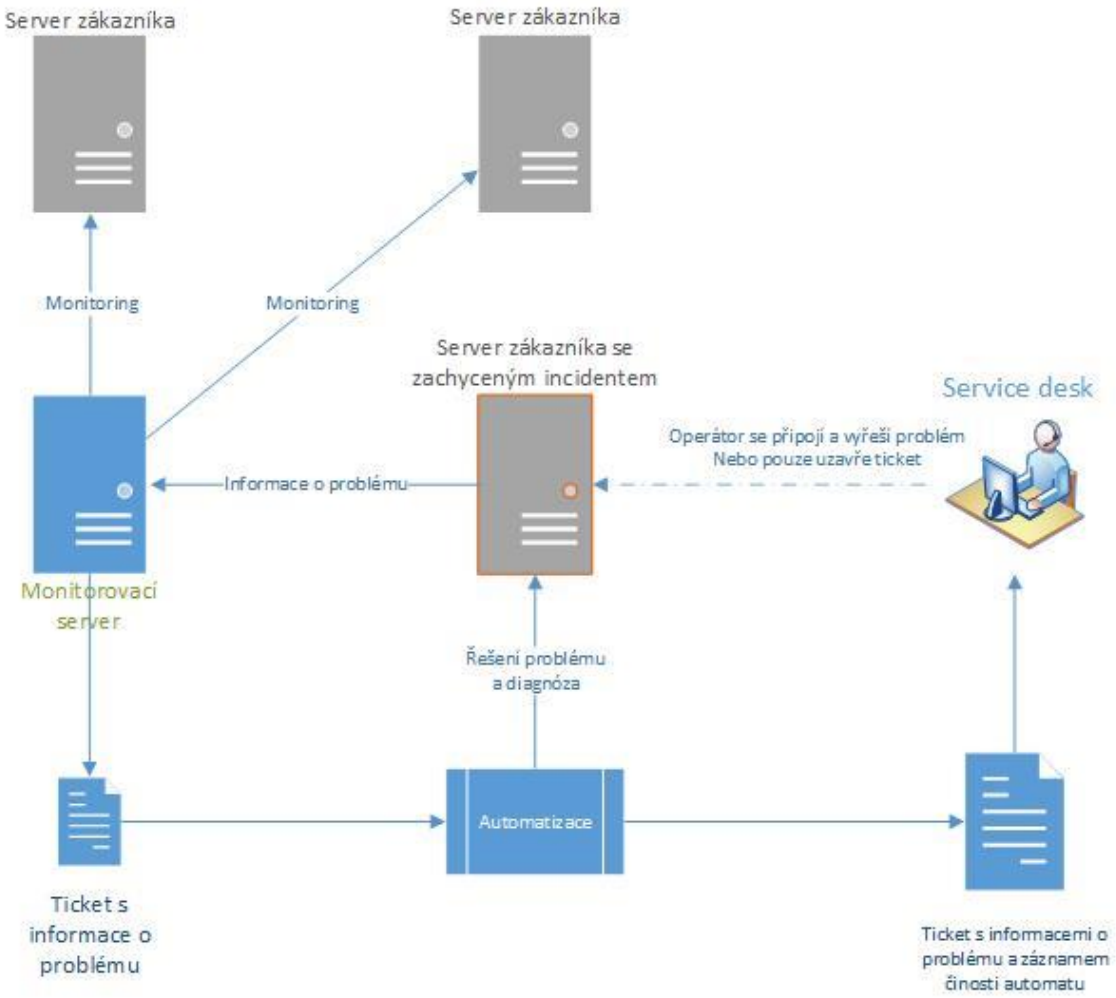
problému vytvořila ticket, který obsahoval popis zachyceného problému. Ten se následně odeslal na centrum service desku, kde se manuálně připojil lidský operátor na server a problém se pokusil vzdáleně vyřešit. Po vyřešení problému byl ticket uzavřen.



Obrázek 3: Správa serverů před automatizací

2.2.1.2 Proces po automatizaci

Po zavedení automatizace se tento proces zásadně změnil. Po zachycení problému monitorovací službou, se vytvořený ticket nejprve odešle na zpracování do automatizačního centra, jehož automaty se pokusí problém vyřešit nebo alespoň provést dodatečnou analýzu. Tím ušetří čas operátorovi zákaznické podpory a tak i náklady všech zúčastněných stran. Pokud neexistuje automat, který by rozpoznal daný problém, je ticket přeposlán rovnou na service desk.



Obrázek 4: Správa serverů s automatizací

2.2.2 Automatizační proces

V předchozí části byl proces automatizace zasazen do kontextu. Nyní bude podrobněji představen samotný proces automatizace.

2.2.2.1 Entity v procesu automatizace

Prvním krokem k pochopení principu automatizačního procesu je seznámení s entitami, které se v procesu vykytují:

Server

Servery, které si u společnosti pronajali zákazníci a jsou jim poskytovány jako služba.

Monitorovací server

Nad skupinou spravovaných serverů je zřízen monitorovací server, který vzdáleně zachytává incidenty na spravovaných serverech. Pokud zachytí problém, vytvoří ticket a automaticky ho odešle do společnosti XY k vyřešení.

Ticket

Tickety obsahují informace o problému, který se vyskytuje na daném serveru. Z pohledu automatizace je klíčovým atributem ticketu je tzv. **Alert key**, na základě kterého systém pozná, který automat má daný ticket řešit.

Automat

Automat je v tomto případě program, který je automaticky volán na základě zachycení události na zákaznickém serveru. **Matcher** je klíčovým atributem automatu, pomocí kterého automat pozná, že se má pokusit vyřešit daný ticket. Matcher je možné chápat, jako zámek, který hlídá aby se automat nespustil bez příčiny (vhodného Alert key). Automaty je možné rozdělit na dvě kategorie - remediační a diagnostické.

Remediačními automaty je možné označit ty automaty, jejichž hlavním účelem je jít na zákaznický server a opravit daný problém. Na druhou stranu hlavní účel **diagnostických automatů** je provést diagnózu a tak usnadnit práci service desku, který následný problém řeší.

Do tohoto rozdělení není možné zařadit dva klíčové automaty – automat Escalation Handler a automat CTK. **Escalation Handler** hraje klíčovou roli v procesu automatizace zejména při přiřazování ticketů a příslušných automatů. Escalation Handler reaguje na každý ticket, který přišel do automatizačního procesu. **CTK automat** má za úkol periodicky ověřovat dostupnost zákaznických serverů, čímž je možné předejít problémům s připojením.

Exekuce

Exekuce v tomto případě znamená, že se automat pokusil vykonat svojí činnost. Exekuce mohou mít následující výsledky:

Resolved – exekuce, kde automat dokázal vyřešit problém, který vznikl na straně serveru. Po ukončení práce automat odešle původní ticket a záznam o své činnosti na service desk.

Assisted – exekuce automatů, kde došlo k získání dalších informací o problému na serveru, ale automat sám nebyl schopný tento problém vyřešit. Tyto informace jsou přidány k ticketu, který následně vyřeší service desk.

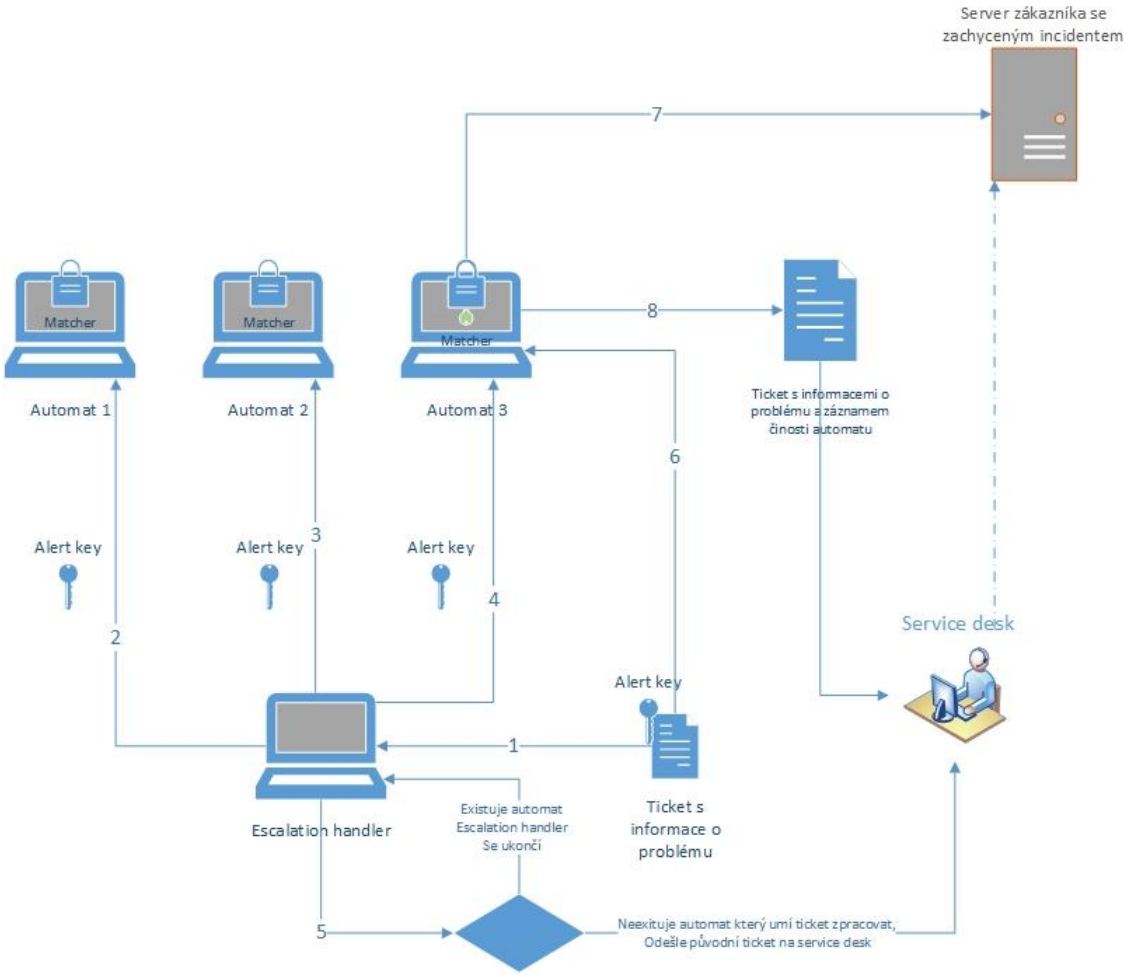
Escalated – exekuce automatu Escalation Handler, které skončily ve spodní větvi rozhodovací stromu na obrázku č. 5. Důvodem pro to může být:

- na daný Alert key zatím neexsituje automat, který by ho dokázal vyřešit,
- daný klient nemá implementovaný automat, který by ho dokázal vyřešit,
- automat, který by měl daný Alert key řešit, je špatně nastavený.

Failed to connect – exekuce, které neuspěly kvůli tomu, že se automat nebyl schopný připojit na zákaznický server.

Failed tickets – označení pro exekuce, které z neznámého důvodu nedokončily svou činnost. Podíl těchto exekucí je za normálního provozu menší než 1%.

2.2.2.2 Proces automatizace



Obrázek 5: Automatizační proces

1) Podproces automatizace začíná tak, že automat Escalation Handler zachytí ticket a jeho Alert key.
2) Následně zkouší, zda má klient automat, který tento proces dokáže vyřešit.
    a. Pokud existuje automat, který dokáže daný Alert key řešit, escalation handler se ukončí.
    b. Pokud neexistuje, odešle původní ticket na service desk, kde ho řeší operátor manuálně, a proces automatizace zde končí.
3) Automat, který dokáže daný problém vyřešit, si vezme data z původního ticketu.
4) Připojí se na server zákazníka, kde se pokusí vyřešit daný incident.
5) Odešle upravenou verzi na service desk, kde operátor pouze zkontroluje činnost automatu a uzavře původní ticket.

2.3 Reportovací aplikace

V následující části bude popsána reportovací aplikace, která shromažďuje data o činnosti automatů a prezentuje je automatizačnímu týmu, zákazníkovi, projektovým manažerů i vrcholovému managementu společnosti.

2.3.1 Datové struktury

Datová struktura aplikace je datový sklad, postavený na relačním databázovém enginu Maria DB. Pro rychlejší načítání reportů není databáze ve 3. normální formě.

2.3.1.1 AutomataExecutions

V této tabulce jsou uloženy záznamy o exekucích.

- **RadarTicket -** int(11) – číslo ticketu v automatizačním týmu
- **RadarStatus –** varchar(20) – v jakém stavu se tento ticket nachází
- **CreatedDate -** datetime – čas vytvoření exekuce
- **CloseDate** - datetime – čas uzavření exekuce
- **ExecutionId** - int(11) – identifikační číslo exekuce, součást primárního klíče
- **InstanceId** – číslo instance automatizačního řešení
- **AutomatonId** int(11) – identifikační číslo automatu, který provedl exekuci
- **Automaton** - varchar(255) – název automatu, který provedl exekuci
- **AutomataType** - varchar(50) – zda se jedná o remediační nebo diagnostický automat
- **Description** - varchar(200) – základní popis problém z původního ticketu
- **InstanceValue** - varchar(200) – nadstavba AlertKey, v případě některých automatů, matcher vyžaduje kromě odpovídajícího AlertKey i odpovídající instance value
- **AlertKey** - varchar(100) – hodnota alert key ticketu
- **AlertGroup** - varchar(50) – do jaké skupiny Alert Key patří
- **Summary** - varchar(512) – shrnutí problému, jedná se o součást ticektu
- **ExecutionMode** - varchar(50) – zda byla exekuce spuštěna ručně, nebo automaticky
- **TicketOutcome** - varchar(20) – stav ticketu (remediation, escalation nebo diagnosis)

- **FailedToConnect** - varchar(1) – selhal automat už při připojení na server
- **ExecutionStatus** - varchar(25) – jeden z klíčových atributů pro reportovací nástroj, určuje, výsledek exekuce (remediation, escalation, assited, failed to conncet, failed)
- **Status** - varchar(20) – určuje výsledek činnosti automatu
- **eBondTicket** - varchar(20) – číslo ticketu z monitorovací služby
- **IMT** - varchar(10) – logická skupina, do které zákazník patří
- **ClientTriname** - varchar(3) – jednoznačné třípísmené označení klienta
- **ClientName** - varchar(100) – plný název klienta
- **ClientID** - int(11) – identifikační číslo klienta
- **Hostname** - varchar(100) – název serveru, na kterým byla provedena exekuce
- **LastModification** - timestamp – datum a čas poslení úpravy záznamu

2.3.1.2 CmdbDevices

Tato tabulka slouží jako databáze uživatelských serverů.

- **DeviceId** – int(11) – identifikační číslo zařízení
- **DeviceType** - varchar(15) – druh zařízení (server/virtuální server)
- **DeviceName** - varchar(100) – název zařízení
- **InstanceId** - tinyint(4) – číslo instance (viz. AutomataExecutions)
- **ImtId** - tinyint(4) – identifikační číslo logické skupiny
- **IMT** - varchar(50) – logická skupina, do které patří zákazník
- **ClientId** - smallint(6) – identifikační číslo zákazníka
- **ClientTriname** - varchar(5) – jednoznačné třípísmené označení klienta
- **ClientName** - varchar(100) – plný název klienta
- **MonitoredAddress** – varchar(16) – IP adresa server
- **Description** - varchar(256) – popis zařízení
- **OsType** - varchar(20) – jaký operační systém je na zařízení používán
- **Tier** - varchar(20) – skupina, do které server patří (např. produkční)
- **Notes** - varchar(256) – poznámka
- **AssetId** - varchar(50) – identifikační číslo z pohledu monitoringu
- **LocatedAtSite** - varchar(100) – informace o fyzické umístěné serveru

- **IPCenterManaged** - char(5) – zda je server kompletně spravován společnosti XY
- **Monitored** - char(5) – zda je pravidelně monitorován
- **Status** - varchar(100) – jaký má status (odpojený, aktivní)
- **SnapshotDate** – date – pro které datum, je tento stav platný
- **LastModification** - timestamp – datum a čas poslední úpravy záznamu

2.3.1.3 Dashboard

Předpočítaná tabulka, která slouží jako zdroj informací pro dashboard aplikace.

- **ClientTriname** - varchar(5) – jednoznačné třípísmené označení klienta
- **ClientId** - smallint(6) – identifikační číslo zákazníka
- **Date** - date – datum, pro který je tento záznam platný
- **AutomataRemediation** – smallint(6) – počet aktivních remediačních automatů
- **AutomataDiagnosis** - smallint(6) – počet aktivních diagnostických automatů
- **ActiveCI** - mediumint(9) – počet aktivních serverů z tabulky CmdbDevices
- **Connectivity** - smallint(6) – výsledek CTK testů
- **IncidentResolved** - mediumint(9) – počet „Resolved“ exekucí
- **IncidentAssisted** - mediumint(9) – počet „Assisted“ exekucí
- **IncidentEscalation** - mediumint(9) – počet „Escalation“ exekucí
- **IncidentFailedToConnect** - mediumint(9) – počet „FailedToConnect“ exekucí
- **IncidentFailed** - mediumint(9) – počet „Failed“ exekucí
- **RadarTickets** - mediumint(9) – počet ticketů zpracovaných za den
- **LastCTKCheck** - varchar(256) – odkaz na výsledky posledního CTK testu
- **LastCTKDate** - date – datum posledního CTK testu
- **LastModfication** - timestamp – poslední modifikace záznamu

2.3.1.4 LiveAccounts

Seznam zákazníků.

- **ImtId** - tinyint(4) – identifikační číslo logické skupiny
- **IMT** - varchar(50) – logická skupina, do které patří zákazník
- **ClientId** - smallint(6) – identifikační číslo zákazníka
- **ClientTriname** - varchar(5) – jednoznačné třípísmené označení klienta
- **ClientName** - varchar(100) – celý název klienta

- **InstanceId** - tinyint(4) – číslo instance
- **CountryID** - smallint(6) – identifikační číslo země
- **Country** - varchar(100) - název země
- **GoLive** - date - datum zavedení klienta do automatizační procesu
- **SunSet** - date - datum ukončení klienta v automatizační procesu
- **AccountStatus** - varchar(15) – stav klienta
- **WikiUid** - varchar(50) – odkaz na informace o klientovi
- **LastModification** - timestamp – datum a čas poslední modifikace záznamu

2.3.1.5 LiveAutomas

Seznam právě nasazených automatů

- **ImtId** - tinyint(4) – identifikační číslo logické skupiny
- **IMT** - varchar(50) – logická skupina do které patří zákazník
- **ClientId** - smallint(6) – identifikační číslo zákazníka
- **ClientTriname** - varchar(5) – jednoznačné třípísmené označení klienta
- **ClientName** - varchar(100) – plný název klienta
- **InstanceId** - tinyint(4) – číslo instance (viz. AutomataExecutions)
- **AutomatonID** - bigint(20) – identifikační číslo automatu
- **Automaton** - varchar(255) –plný název automatu
    - Pravidla pojmenovávání automatu: Systém:R/D:Název:v1.2.3.4
- **Purpose** - varchar(11) – určuje, zda je automat remediační nebo diagnostický
- **AutomatonGoLive** - date – datum spuštění první verze automatu
- **AutomatonLastApproval** - date – datum spuštění aktuální verze automatu
- **Matchers** – text – regulární výraz který je porovnáván s AlertKey, na základě výsledku porování se spustí automat
- **AutoCommonName** - varchar(100) – pouze název automatu
- **LastModification** - timestamp – datum a čas poslední modifikace záznamu

2.3.1.6 Předpočítané tabulky

- **AccountOverview** – databázová struktura kopíruje reportu Account overview. Jsou to agregovaná data podle klientů z výše uvedených tabulek.

- **AutomataOverview** – databázová struktura kopíruje reportu Account overview. Jsou to agregovaná data podle automatů z výše uvedených tabulek.
- **EventServers** – agregovaná data o počtu incidentů podle serverů.

2.3.2 Analýza použitých technologií

V následující sekci práce budou představeny technologie, na kterých je aplikace postavena. Vzhledem k tomu, že je nabízené řešení distribuováno do ostatních geografických lokalit, byl výběr ovlivněn následujícími faktory:

- požadavky na snadnou konfiguraci,
- nízké až nulové náklady na prerekvizity,
- technologie musí být schválené společností XY.

2.3.2.1 Databáze MariaDB

MariaDB je relační databáze, která se odštěpila (je tzv. Forkem) od známější databáze MySQL. Hlavním důvodem k vytvoření této větve bylo udržení licence svobodného softwaru GNU GPL. Iniciativa, díky které tato větev vznikla, pochází od původních vývojářů MySQL, kteří se obávali o další směřování tohoto softwaru po jeho odkoupení společností Oracle. [23]

2.3.2.2 ETL proces

Na severech jsou pomocí crontabu (několikrát za den) spouštěny synchronizační skripty psané v jazyce PHP, které pravidelně aktualizují data v reportovací aplikaci. Při transformačním procesu dochází k agregování dat pro urychlení běhu aplikace.

2.3.2.3 Webové rozhraní

Backend samotné webové aplikace je vytvořen za pomocí ZEND Framework a jazyka PHP. Frontend aplikace využívá nástroj Gentellela Admin, ve kterém se nachází knihovny pro moderní grafy, tabulky a formuláře.

2.3.3 Uživatelská analýza

V následující části představím podrobně nástroj z uživatelského pohledu. Aplikaci lze z uživatelského pohledu rozdělit do dvou logických celků:

- dashboard,
- reporty.

2.3.3.1 Dashboard

Dashboard má za úkol poskytnout uživateli rychlý přehled o exekucích automatů na daných klientech.

Současný stav:

V současné systému dává dashboard pouze základní přehled o číslech, ale nedává zákazníkům aplikace téměř žádné analytické možnosti. Pro větší analýzu musí uživatel stahovat report exekucí a jeho následnou analýzou v softwaru třetí strany získat potřebné informace.

V horní části dashboardu aplikace, dostává uživatel přehled o základních ukazatelech a jejich změnou za poslední 2 týdny. Tyto ukazatele jsou:

- celkový počet automatů,
- počet remediačních automatů,
- počet diagnostických automatů,
- počet aktivních serverů,
- procento konektivity na tyto servery,
- počet incidentů na jeden server za poslední 2 týdny.

Následuje graf, který zobrazuje průběh exekucí za posledních 14 dní. Tyto informace jsou dostupné buď pro všechny klienty, vybranou oblast nebo jednotlivého klienta.

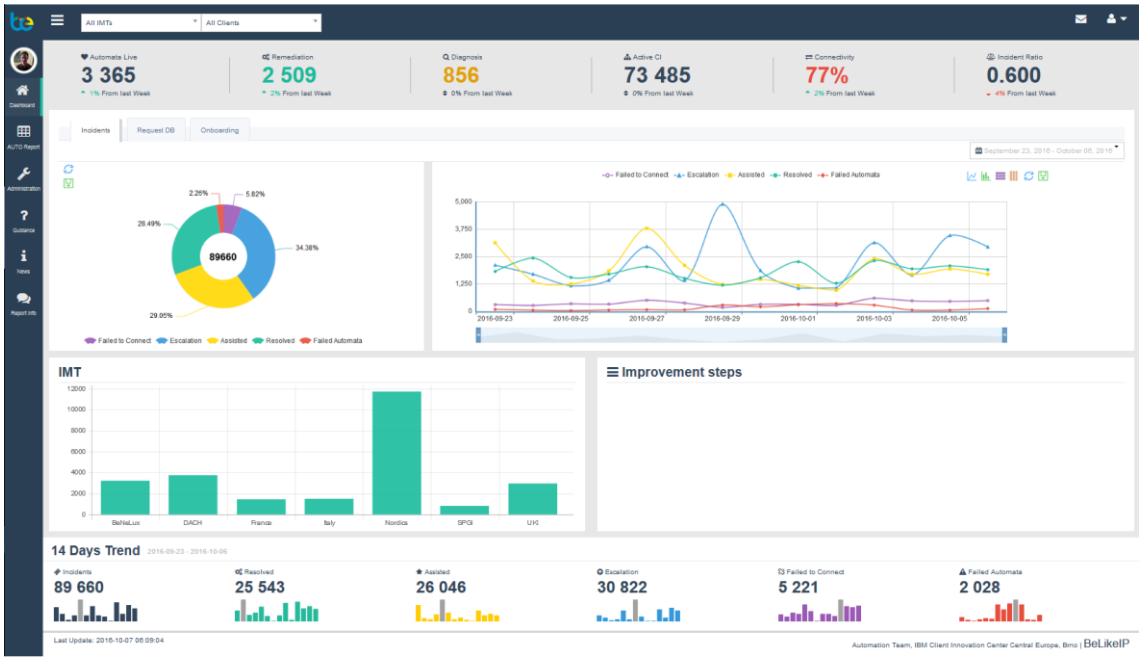
Obrázek 6: Dashboard 2.0 (testovací data)

Chystaná verze dashboardu:

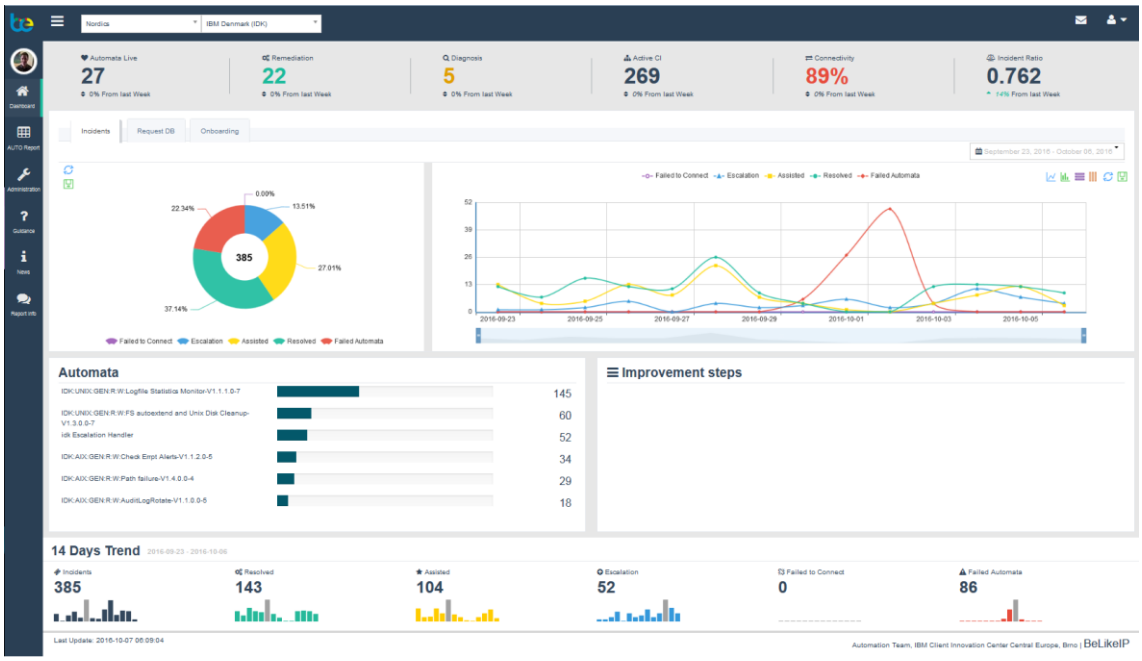
V chystané verzi aplikace je hlavním cílem přizpůsobit dashboard tak, aby přehledně poskytl uživatelům informace, které získávali dodatečnou analýzou. Sloužit k tomu má nová sekce sloupcových grafů v kombinaci se sekcí „highlights“, jejichž návrh bude tvořit hlavní část této práce.

Sloupcový graf v nové verzi dashboardu ukazuje uživateli následující informace:

- globální sekce – exekuce vybraného druhu podle jednotlivých oblastí,
- úroveň skupiny – exekuce vybraného druhu podle klientů ve vybrané skupině,
- klientská úroveň – rozdělení exekucí na jednotlivé automaty.



Obrázek 7: Dashboard 3.0 skupina (testovací data)



Obrázek 8: Dashboard 3.0 – klient (testovací data)

2.3.3.2 Reporty

Reporty dávají uživateli všechna dostupná data v uspořádaných tabulkách s možností jejich stažení v různých formátech. Vzhledem k tomu, že se tato práce zbývá převážně sekcí dashboard, následuje pouze stručný přehled o obsahu jednotlivých reportů nástroje:

Account overview – přehled základních informace rozdělených podle klientů.

Automata overview - přehled základních informací rozdělených podle automatů.

Execution overview – výpis exekucí se všemi dostupnými informacemi.

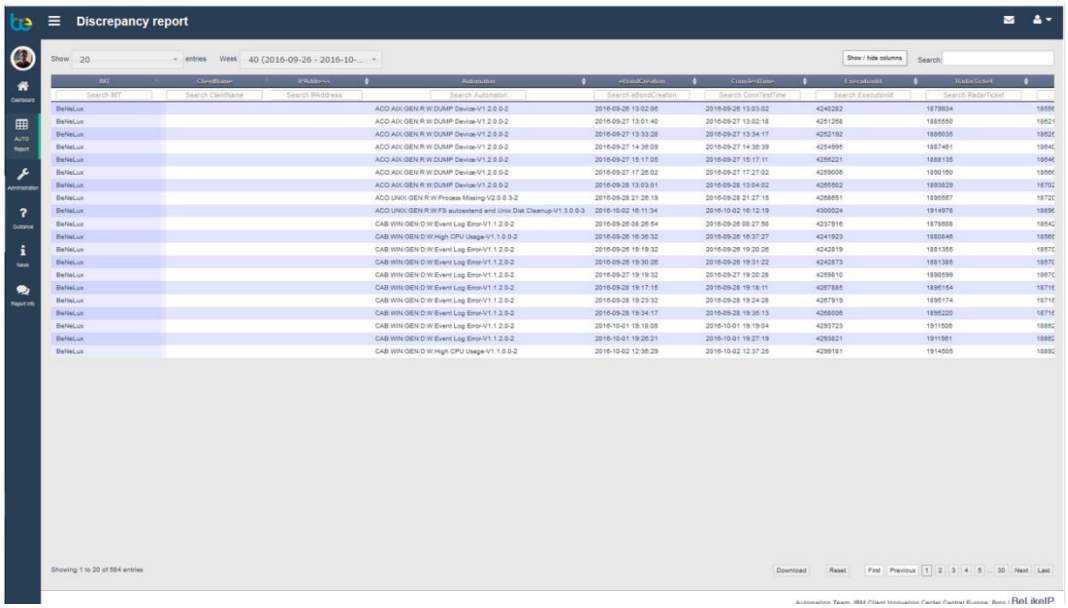
Onboarding report – přehled testovacích exekucí, používáno při nasazování automatizace pro nového klienta.

Weekly status – přehled základních informací rozdělených podle času.

Cmdb catalog – katalog serverů, na kterých běží automaty.

Events servers – zobrazuje počet problémů na zákaznických serverů.

Discrepancy report – databáze exekucí, které zjistily špatné nastavení serveru v CMDB katalogu.



Obrázek 9: 3.0 - ukázka reportu

2.4 Řešitelné slabé stránky systému

V následující části budou shrnuty vybrané slabé stránky aplikace, které vyplývají z vlastní analýzy aplikace a z konzultace s analytickým oddělením společnosti.

Problém s navigací

Při zobrazení skupiny uživatel vidí na grafu nárůst ve druhu exekucí, které právě analyzuje. Problém nastává v případě, že chce konkrétně vědět, který klient ze zobrazené skupiny tento nárůst zapříčinil. V chystané verzi systému musí manuálně „zkoušet“, na kterém klientu z vybrané skupiny najde stejný nárůst.

Zobrazeny problémy, ne řešení

Z dashboardu uživatel vidí, že nemá optimální čísla. Program mu ale neporadí, co má dělat, aby daná čísla zlepšil.

Jeden úhel pohledu na data

Současná verze dashboardu poskytuje pohled na exekuce pro jednotlivé klienty. Zajímavou informací by pro mnohé uživatele byl pohled z hlediska automatů, serverů nebo alert key.

Vysoký počet eskalovaných ticketů

Eskalované exekuce jsou jedním z nežádoucích výsledků činnosti automatu.

Chyby v CMDB katalogu

V CMDB katalogu se objevují elementární chyby, které zvyšují počet failed to conncet exekucí.

3 NÁVRH VLASTNÍHO ŘEŠENÍ

V části návrhu vlastního řešení se budu zaměřovat na vybrané problémy nové verze a vytvořím návrh jejich řešení, které bude zároveň sloužit jako zadání front-end programátorům analyzovaného nástroje.

3.1 Koncepce sekce „Highlights“

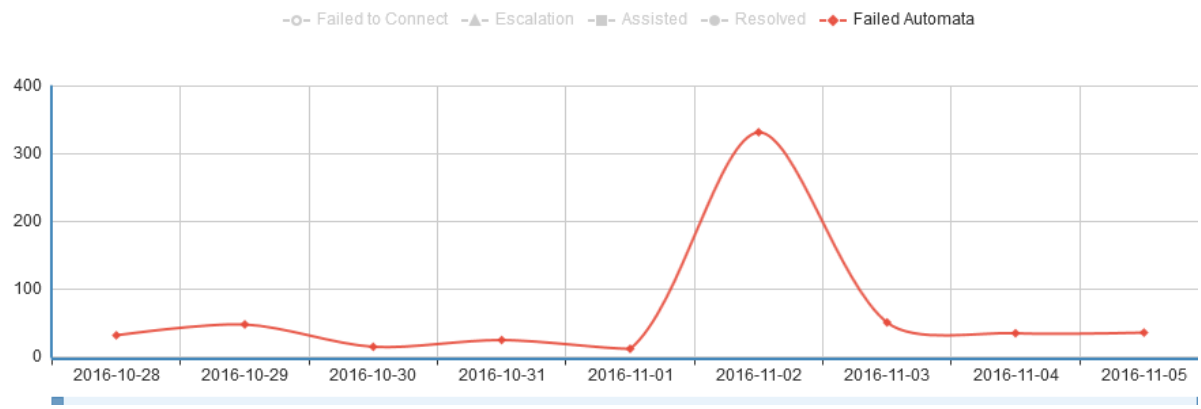
Sekce highlights by měla uživatele apliakce upozorňovat na problémy, které se mohou objevovat u jednotlivých klientů. Měla by mu dát informace, které nelze z chystané verze dashboardu vyčíst. Pokud to bude možné, měly by uživatele navést, jak detekovaný problém řešit.

3.2 Analýza odchylek

První typ zpráv do sekce highlights je analýza odchylek, která dává uživateli informace o zajímavých bodech grafu, které nejsou z grafu čitelné.

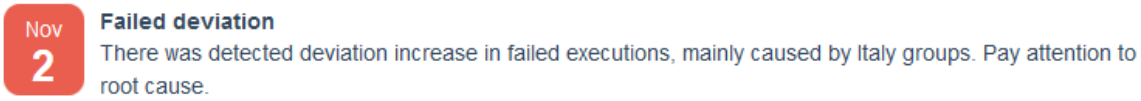
3.2.1 Příklad využití:

Uživatel si zobrazí skupinu klientů a uvidí v grafu chybných exekucí výkyv:



Obrázek 10: Analýza odchylek - exekuce

Toto ho upozorní na fakt, že na některém z klientů, které si právě zobrazuje se stalo něco neočekáváného, ale pokud by chtěl provádět zákazník další analýzu (např. zjistit na kterém konkrétním klientu došlo k tomuto nárůstu), musel by nejprve ručně otevírat jednotlivé klienty a hledat, kde bude na grafu podobný skok jako na grafu vybrané skupiny. Analýza odchylek bude zákazníka tímto procesem navigovat.



3.2.2 Princip detekování odchylek

Před volbou vhodné statistické metody pro detekování odchylek je nutné stanovit požadavky na očekávané výsledky:

- metoda musí reagovat na dlouhodobý vývoj klienta a nevyhodnocovat dlouhodobý nárůst jako odchylku,
- součet klientů ve skupině musí dát dohromady skupinu,
- metoda nesmí být složitá na výpočet.

Po analýze provedené na reálných datech bylo zvoleno následující řešení:

1) Spočítání běžné hladiny (CL) exekucí u klienta

$$\mathrm{CL} = 0{,}75 * \mathrm{median}(\text{vybraného období}) + 0{,}25 * \mathrm{průměr}(\text{vybraného období})$$

Metoda v sobě kombinuje výhody mediánu i aritmetického průměru.

2) Vypočítání horní (UCL) a dolní (LCL) hranice

$$\mathrm{UCL} = \mathrm{CL} * 1{,}5 + 10$$

$$\mathrm{LCL} = \mathrm{CL} * 0{,}6 - 10$$

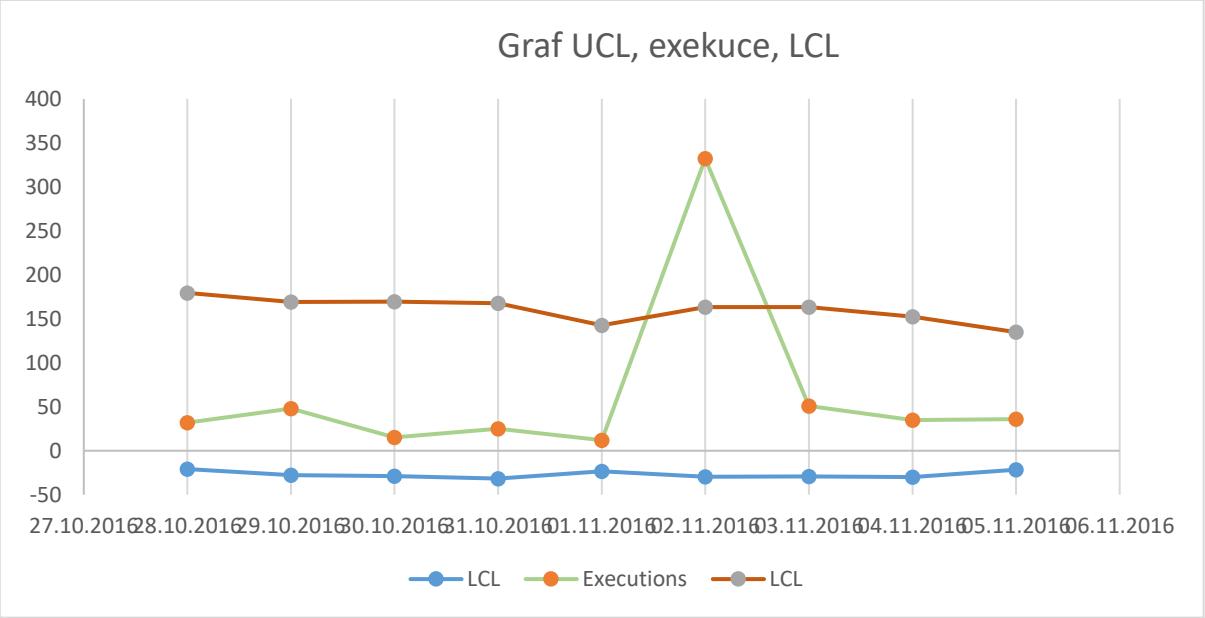
Tato metoda byla zvolena převážně pro svou jednoduchost a rychlý výpočet. Hodnoty +- 10 jsou zde kvůli klientům, kteří mají malý počet exekucí. Díky těmto hodnotám nebude považována za výkyv situace, kde by se střední hodnota CL blížila nule, a v jeden den by přišly 2 exekuce, což je považováno za zanedbatelnou hodnotu.

V ideálním případě by do těchto vzorců byla promítnuta i směrodatná odchylka těchto grafů tak, aby méně stabilní klienti měli více benevolentní hranice detekce, ale její zakomponování by tesotvací dotazy významně zpomalilo.

3) Porovnání počtu exekucí za daný den s hranicemi UCL a LCL

Pokud je počet exekucí za daný den větší než hranice UCL jedná se o odchylku zvýšení, pokud je počet exekucí nižší než LCL, jedná se o odchylku snížení.

Způsob detekce byl inspirován regulačními diagramy, které jsou popsány v teoretické části.



Obrázek 11: Princip detekce odchylek

3.2.3 Příprava dat

První krokem v realizaci tohoto řešení je příprava databázové struktury.

3.2.3.1 Databázová struktura

Z důvodu snížení zátěže serveru a zachování plynulosti programu je nutné některé hodnoty předpočítávat pomocí synchronizačních skriptů a ukládat je do databázové tabulky:

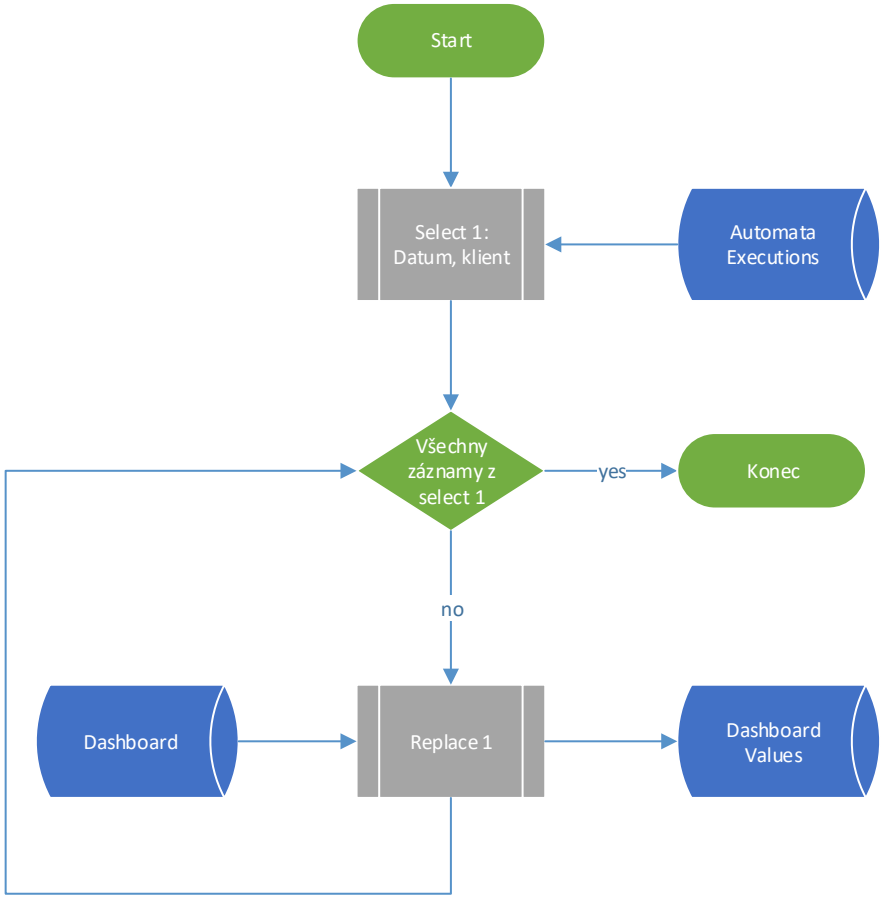
DashboardValues:

- **Datum –** datum, pro které platí hodnoty UCL a LCL
- **Klienti -** klient, pro kterého platí hodnoty UCL a LCL
- **UCL hodnoty** – celkem 5 sloupců, jeden pro každý druh výsledku exekuce.
- **LCL hodnoty –** celkem 5 sloupců, jeden pro každý druh výsledku exekuce.

```
CREATE TABLE `DashboardValues` (
	`Date` DATE NOT NULL,
	`Client` VARCHAR(3) NOT NULL,
	`ResLCL` DECIMAL(10,4) NULL DEFAULT NULL,
	`ResUCL` DECIMAL(10,4) NULL DEFAULT NULL,
	`AssLCL` DECIMAL(10,4) NULL DEFAULT NULL,
	`AssUCL` DECIMAL(10,4) NULL DEFAULT NULL,
	`EscLCL` DECIMAL(10,4) NULL DEFAULT NULL,
	`EscUCL` DECIMAL(10,4) NULL DEFAULT NULL,
	`FtcLCL` DECIMAL(10,4) NULL DEFAULT NULL,
	`FtcUCL` DECIMAL(10,4) NULL DEFAULT NULL,
	`FaiLCL` DECIMAL(10,4) NULL DEFAULT NULL,
	`FaiUCL` DECIMAL(10,4) NULL DEFAULT NULL,
	`LastModification` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
	PRIMARY KEY (`Date`, `Client`)
)
COLLATE='utf8_general_ci'
ENGINE=MyISAM;
```

3.2.3.2 Synchronizační skript

Synchronizační skript bude automaticky spouštěn každý den v 5:00 (před pracovní dobou, ale v době kdy jsou v databázi kompletní informace za předchozí den). Pro každého klienta se každý den uloží do tabulky DashboardValues hodnota UCL a LCL. Při vypisování highlights se bude porovnávat hodnota z tabulky dashboard s uloženou hodnotou.



Obrázek 12: Synchronizační skript - DashboardValues

Select 1:

```
SELECT
   DB.ClientTriname Triname,
   DB.Date Datum
FROM
   IPadministration.Dashboard DB
WHERE
   DB.Date >= DATE_SUB(CURRENT_DATE,  INTERVAL 7 DAY) AND
   DB.Date <> CURRENT_DATE
```

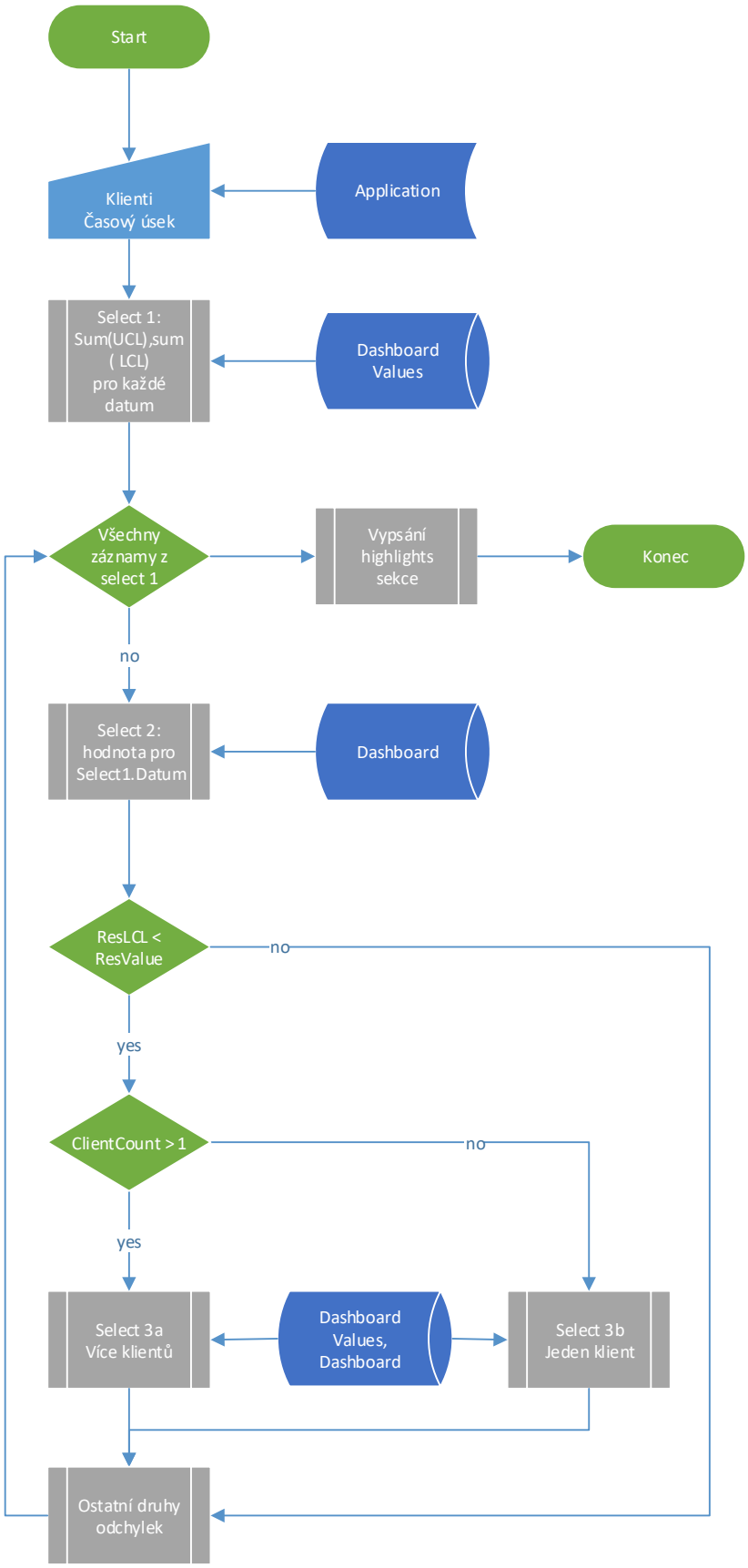
Replace 1:

```
REPLACE INTO IPadministration.DashboardValues2
(Date,`Client`,ResLCL,ResUCL,AssLCL,..., FaiUCL)
SELECT
   save.Date,
   save.`Client`,
   save.ResLCL,
   save.ResUCL,
   save.AssLCL,
   ...
   save.FaiUCL
FROM (
   SELECT
```

```
      Select1.Datum AS Date,
      Select1.Triname AS Client,
      #ResCL = 0.25 * 14 days AVG + 0.75 * 14 days MED
      @ResCL:=0.25*
         (
         SELECT
            AVG(DB.IncidentResolved)
         FROM
            IPadministration.Dashboard DB
         WHERE
            DB.Date >= DATE_SUB(Select1.Datum, INTERVAL 14 DAY) AND
            DB.Date <= Select1.Datum AND
            DB.ClientTriname = Select1.Triname
         )
         + 0.75 *
         (  SELECT
               MIN(COL)
            FROM
            (
               SELECT
                  DB.IncidentResolved COL
               FROM
                  IPadministration.Dashboard DB
               WHERE
                  DB.Date >= DATE_SUB Select1.Datum, INTERVAL 14 DAY) AND
                  DB.Date <= Select1.Datum AND
                  DB.ClientTriname = Select1.Triname
               ORDER BY
                  DB.IncidentResolved desc
               LIMIT 8
            )sub_med
         ) AS ResCL,
      (@ResCL*0.60 - 10) ResLCL,
      (@ResCL*1.50 + 10) ResUCL,
      #STEJNĚ SE URČÍ HODNOTY PRO OSTATNÍ TYPY EXEKUCÍ
   FROM
      IPadministration.Dashboard DB
   WHERE
      DB.Date = Select1.Datum and
      DB.Triname = Select1.Triname
)save
```

3.2.4 Návrh implementace ve webovém rozhraní

Prvním krokem v tomto skirptu je získání časového rozptylu a seznam klinetů, pro které budou odchylky vypsány. Ve druhém kroku budou sečteny horní a spodní hranice pro každý den, pro vybranou skupinu klientů (pro každý den vznikne jeden řádek). Výsledek příkazu procházen cyklem, který v každé své iteraci spočítá celkovou hodnotu exekucí pro daný den (Select 2) a porovná jí s horní a spodní hranicí ze Select 1. Pokud bude hodnota ve vyznačených hranicích, skript nic nedělá. Pokud hodnota přesáhne vymezenou hranici, vypíše se upozornění – Select 3.



Obrázek 13: Dashboard - odchylky - webové rozhraní

Select 1:

```
SELECT
   dbv.Date,
   SUM(dbv.ResLCL) + (10 * COUNT (dbv.`Client`)) - 10 AS 'ResLCL',
   SUM(dbv.ResUCL) - (10 * COUNT (dbv.`Client`)) + 10 AS 'ResUCL',
   SUM(dbv.AssLCL) + (10 * COUNT (dbv.`Client`)) - 10 AS 'AssLCL',
   SUM(dbv.AssUCL) - (10 * COUNT (dbv.`Client`)) + 10 AS 'AssUCL',
   SUM(dbv.EscLCL) + (10 * COUNT (dbv.`Client`)) - 10 AS 'EscLCL',
   SUM(dbv.EscUCL) - (10 * COUNT (dbv.`Client`)) + 10 AS 'EscUCL',
   SUM(dbv.FtcLCL) + (10 * COUNT (dbv.`Client`)) - 10 AS 'FtcLCL',
   SUM(dbv.FtcUCL) - (10 * COUNT (dbv.`Client`)) + 10 AS 'FtcUCL',
   SUM(dbv.FaiLCL) + (10 * COUNT (dbv.`Client`)) - 10 AS 'FaiLCL',
   SUM(dbv.FaiUCL) - (10 * COUNT (dbv.`Client`)) + 10 AS 'FaiUCL',
   COUNT(dbv.`Client`) AS 'pocet'
FROM
   IPadministration.DashboardValues2 dbv
WHERE
   dbv.Date >= ''Applikace.DatumOd' AND
   dbv.Date <= 'Applikace.DatumDo' AND
   dbv.`Client` IN (Applikace.Klienti)
GROUP BY
   dbv.Date
```

Select 2:

```
SELECT
   SUM(dbv.IncidentResolved) as 'ResValue',
   SUM(dbv.IncidentAssisted) as 'AssValue',
   SUM(dbv.IncidentEscalation) as 'EscValue',
   SUM(dbv.IncidentFailedToConnect) as 'FtcValue',
   SUM(dbv.IncidentFailed) as 'FaiValue'
FROM
   IPadministration.Dashboard dbv
WHERE
   dbv.Date = Select1.Date AND
   dbv.`ClientTriname` IN (Select1.Klienti)
GROUP BY
   dbv.Date
```

Select 3 – více klientů, snížení:

```
SELECT
   CONCAT('There was detected deviation decrease in resolved executions,
      mainly caused by ',
      group_concat(res.ClientTriname),'
      clients. Pay attention to root cause.') as Message
FROM
(
   SELECT
      db.ClientTriname
   FROM
      IPadministration.DashboardValues2 dbv
      JOIN IPadministration.Dashboard db
         on db.Date = dbv.Date and db.ClientTriname = dbv.`Client`
   WHERE
      dbv.date = Select1.Date AND
      db.ClientTriname IN (Select1.Klienti) AND
      dbv.ResLCL - db.IncidentResolved >= 0
   ORDER BY
      (dbv.ResLCL - db.IncidentResolved) desc
   LIMIT 4
)res
```

3.3 Řešení eskalovaných ticketů

Následující část bude uživatele navádět jak vyřešit eskalované tickety.

3.3.1 Význam

Pokud Escalation Handler „eskaluje“ ticket na service desk, může to mít následující příčiny:

1. neexistuje automat, který by zachycený AlertKey uměl vyřešit:
    - příležitost na **vývoj** nového automatu
2. existuje automat, který umí vyřešit získaný AlertKey, ale není nasazen na analyzovaném klientu:
    - příležitost na **nasazení** automatu na klienta
3. existuje automat, který umí vyřešit získaný AlertKey, zároveň je nasazený na daném klientu, ale nespustil se:
    - příležitost na zlepšení výsledků pomocí **úpravy konfigurace** automatu.

Pomocí těchto upozornění dostane uživatel jednoduchý návod, jak řešit eskalované tickety, a tak zlepšit výsledky automatizace.

3.3.2 Princip řešení eskalovaných ticketů

Prvním krokem je sestavení tabulky AlertKeyCommonName, která bude spojovat AlertKey a Automat, který ho umí řešit. Druhým krokem bude vytvoření tabulky AlertKeyOpportunity, kde bude spojena informace o počtu eskalovaných ticketů pro AlertKey na klientu v daný den. Tato tabulka bude sloužit jako zdroj dat pro highlights.

3.3.3 Příprava dat

Vytvoření této zprávy vyžaduje dvě zdrojové tabulky:

- AlertKeyCommonName
    - AutomatonCommonName – název automatu
    - AlertKey – alert key
- AlertKeyOpportunity
    - AlertKey – alert key
    - Client - klient
    - SnapshotDate - datum

- Escalated – počet eskalovaných ticketů
- Opportunity – příležitost (nasazení, vývoj, úprava)
- Automaton – který automat umí příležitost řešit

3.3.3.1 AlertKeyCommonName

Ve společnosti v současné době neexistuje ucelený přehled automat – AlertKey, který by byl vhodný pro vytvoření této tabulky. Tabulka bude vytvořena na základě historických dat z tabulky AutomataExecutions, pomocí následujícího příkazu:

```
CREATE TABLE
`AlertKeyCommonName` (
	`AlertKey` VARCHAR(100) NULL DEFAULT NULL,
	`AutomatonCommonName` VARCHAR(100) NULL DEFAULT NULL
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB
;
```

```
SELECT
	MyTable.AutomatonCommonName,
	MyTable.AlertKey
FROM (
	SELECT
		LEFT(AEV.AutomatonVersion, LOCATE('-V',(AEV.AutomatonVersion))-1)
			AS AutomatonCommonName,
		SUBSTR( AEV.AlertKey,5) 'AlertKey'
		#COUNT(*) AS 'SOLVED'
	FROM (
		SELECT
			*,
			RIGHT(AE.Automaton, LOCATE(':',REVERSE(AE.Automaton))-1)
				AS AutomatonVersion
		FROM
			IPadministration.AutomataExecutions AE
		WHERE
			(
				AE.ExecutionStatus = 'Resolved'
				OR AE.ExecutionStatus = 'Assisted'
			)
			AND AE.StartDate >= DATE_SUB(CURRENT_DATE,INTERVAL 90 DAY)
		) AS AEV
	WHERE
		AEV.StartDate >= DATE_SUB(CURRENT_DATE, INTERVAL 90 DAY)
	GROUP BY
		LEFT(AEV.AutomatonVersion,
			LOCATE('-V',(AEV.AutomatonVersion))-1),
		SUBSTR(AEV.AlertKey,5)
	) AS MyTable
```

V některých případech se při vytvoření této tabulky pomocí výše uvedeného příkazu stalo, že jeden AletKey měl k sobě přiřazených více automatů. Tento stav je na první pohled nelogický, ale mohl vzniknout z následujících důvodů:

- v uvedeném období byl automat nahrazen novým automatem, který řeší stejný alert key, ale má jiné jméno,
- automat měl špatně nastavený matcher.

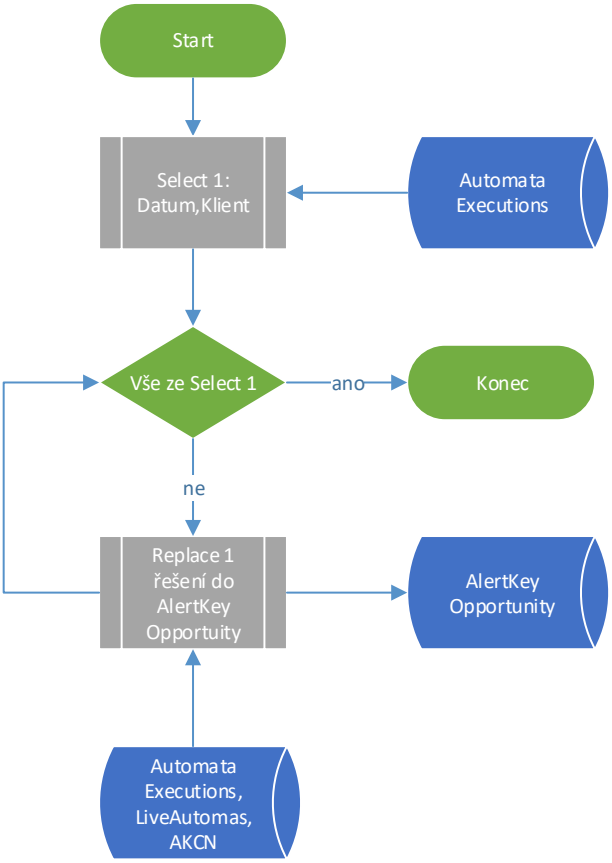
Pro zlepšení výsledků je tedy nutné přidat do původního příkazu `COUNT(*) AS 'SOLVED'`, které nám bude určovat „důležitost“ daného záznamu. Následně budou odstraněny duplicitní hodnoty za využití příkazu join:

```
INSERT INTO IPadministration.AlertKeyCommonName
       (AutomatonCommonName, AlertKey)
SELECT
       MyTable.AutomatonCommonName,
       MyTable.AlertKey
FROM
       MyTable #definovaná výše
JOIN
(
       SELECT
              *,
              max(SOLVED) 'MAX_COUNT'
       FROM
              MyTable
       GROUP BY
              SUBSTR( MyTable.AlertKey,5)
) MyMax
       ON MyMax.AlertKey = MyTable.AlertKey AND
          MyMax.MAX_COUNT = MyTable.SOLVED
```

I přes tato vylepšení musí být tabulka **revidována a aktualizována analytickým týmem**. Tento příkaz tedy bude spuštěn jednorázově na přímo na databázovém serveru. Tabulka zároveň představuje potenciál pro další vylepšení aplikace.

3.3.3.2 AlertKeyOpportunity

Do této tabulky budou pravidelně ukládány ukládány denní výsledky řešení eskalovaných ticketů na všech klientech. Prvním krokem je nalézt všechny klienty, kteří mají v daný den eskalovanou exekuci – Sekect 1. Tyto hodnoty budou složit jako vstup do Replace 1.



Obrázek 14: AlertKeyOpportunity - synchronizace

```
CREATE TABLE
	`AlertKeyOpportunity` (
	`SnapshotDate` DATE NOT NULL,
	`ClientTriname` VARCHAR(3) NOT NULL,
	`AlertKey` VARCHAR(300) NOT NULL,
	`Opportunity` VARCHAR(50) NULL DEFAULT NULL,
	`Automaton` VARCHAR(300) NULL DEFAULT NULL,
	`Escalated` MEDIUMINT(9) NULL DEFAULT NULL,
	`LastModification` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
	`Id` INT(11) NOT NULL AUTO_INCREMENT,
	PRIMARY KEY (`Id`)
)
COMMENT='AlertKey escalated opportunity'
COLLATE='utf8_general_ci'
ENGINE=InnoDB
```

Select 1:

```
SELECT
	AE.ClientTriname,
	AE.StartDate
FROM
	IPadministration.AutomataExecutions AE
```

```
WHERE
       AE.StartDate >= DATE_SUB(CURRENT_DATE, INTERVAL 10 DAY) AND
       AE.ExecutionStatus = 'Escalation'
GROUP BY
       AE.ClientTriname,
       AE.StartDate
```

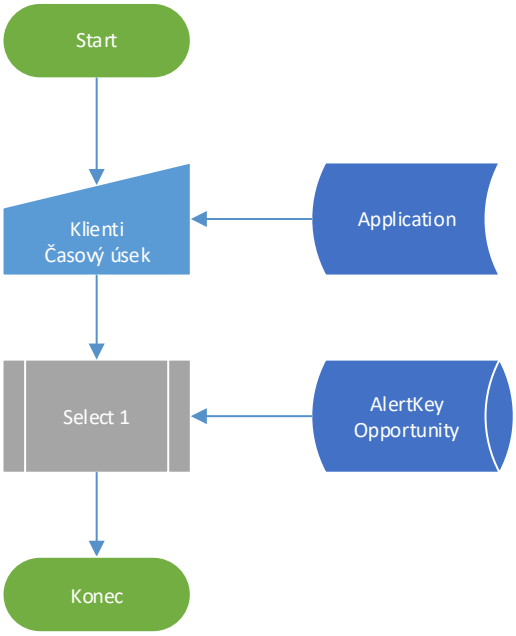
Replace 1:

```
REPLACE INTO AlertKeyOpportunity (SnapshotDate, ClientTriname, AlertKey,
Escalated, Opportunity, Automaton)
SELECT
   AE.StartDate as SnapshotDate,
   AE.ClientTriname,
   AE.AlertKey,
   COUNT(*) 'Escalated',
   CASE
      WHEN (AKCN.AutomatonCommonName IS NULL AND LA.Automaton IS NULL)
         THEN 'Development'
      WHEN (AKCN.AutomatonCommonName IS NOT NULL AND LA.Automaton IS NULL)
         THEN 'Deployment'
      WHEN (AKCN.AutomatonCommonName IS NOT NULL AND LA.Automaton IS NOT NULL)
         THEN 'Adjustment'
   END AS 'Opportunity',
   CASE
      WHEN (AKCN.AutomatonCommonName IS NULL AND LA.Automaton IS NULL)
         THEN ''
      WHEN (AKCN.AutomatonCommonName IS NOT NULL AND LA.Automaton IS NULL)
         THEN AKCN.AutomatonCommonName
      WHEN (AKCN.AutomatonCommonName IS NOT NULL AND LA.Automaton IS NOT NULL)
         THEN AKCN.AutomatonCommonName
   END as Automaton
FROM
   IPadministration.AutomataExecutions AE
   LEFT JOIN IPadministration.AlertKeyCommonName AKCN on
      AE.AlertKey LIKE CONCAT('%',AKCN.AlertKey)
   LEFT JOIN IPadministration.LiveAutomas LA on
      LA.AutoCommonName = AKCN.AutomatonCommonName AND
      LA.ClientShort = AE.ClientTriname
WHERE
   AE.StartDate = 'Select1.Date' AND
   AE.ClientTriname = 'Select1.Client' AND
   AE.ExecutionStatus = 'Escalation'
GROUP BY
   AE.ClientTriname,
   AE.AlertKey
```

Další možné využití této nově získané informace by bylo, rozšíření execution reportu o nové sloupce s těmito hodnotami.

3.3.4 Návrh implementace ve webovém rozhraní

S takto připravenými daty v tabulce AlertKeyOpportunity se k sestavení upozornění pro uživatele použije následující skript:



Obrázek 15: AlertKey opportunity - webové rozhraní

Select 1:

```
SELECT
   *
FROM (
   SELECT
      CONCAT(
         'Deployment of ', AKO.Automaton, ' on ',AKO.ClientTriname,
         ' client could solve up to ', SUM(AKO.Escalated), ' escalated
         executions.') AS 'Message'
   FROM
      IPadministration.AlertKeyOpportunity AKO
   WHERE
      AKO.ClientTriname IN (Application.Klienti) AND
      AKO.SnapshotDate >= 'Application.DatumOd' AND
      AKO.SnapshotDate <= 'Application.DatumDo' AND
      AKO.Opportunity IN ('Deployment')
   GROUP BY
       AKO.Automaton,
       AKO.ClientTriname
   ORDER BY
       SUM(AKO.Escalated) DESC
   LIMIT 1)a
UNION ALL #Developement, Adjustment...
```

3.4 Problémy v nastavení serverů v CMDB katalogu

Katalog zařízení je importován přímo od zákazníka. U některých zákazníků se stává, že jsou v tomto katalogu elementární chyby. To zapříčiňuje problémy automatů, kterým přijde ticket, ale nemohou se na daný server připojit. Tato oblast je dle mého názoru procesně špatně zvládnutá - lepší řešení tohoto problému by bylo implementování větších kontrol už při vkládání těchto informací do databáze. Částečné řešení tohoto problému bude upozorňování na automaticky detekovatelné chyby, které se v tomto katalogu nachází.

3.4.1 Význam a princip

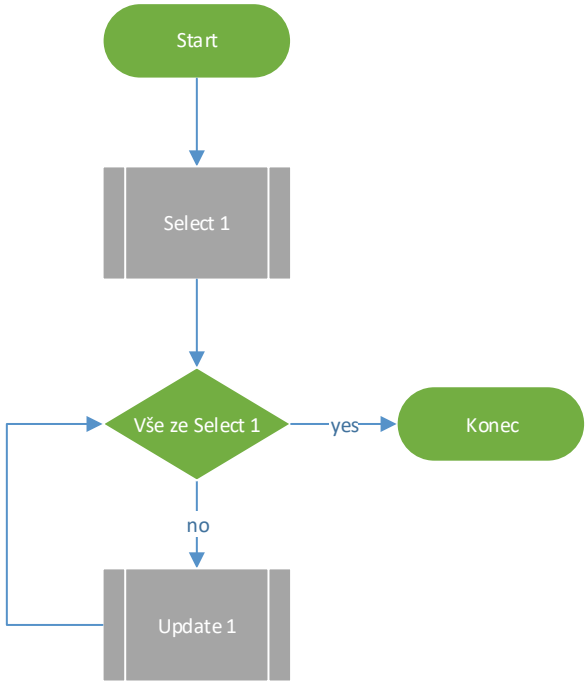
V sekci highlights bude uživatel upozorněn na jakékoliv detekovatelné chyby, které se v CMDB katalogu na vybraných klientech právě vyskytují. Na základě reverzní analýzy katalogu a konzultace s technickými pracovníky společnosti byly sestaveny následující kontroly:

- Kontrola DeviceName - Triname
    - pokud je zařízení server, musí doménový název končit na Triname zákazníka.
- Kontrola DeviceName – IPadresa
    - pokud se zařízení používá k připojení k serveru, musí být doménové jméno IPadresa.
- Kontrola DeviceName – Unikátkost
    - Doménové jméno musí být unikátní.
- Kontrola OSType – Existence
    - Pokud se jedná o server, musí mít definovaný operační systém.
- Kontrola OSType – Description
    - Pokud se jedná o server s Windows, musí mít v popisu nadefinované doménové jméno.
- Kontrola MonitoredAddress
    - MonitoredAddress musí mít tvar IPadresy IPv4.
- Kontrola Status – existence
    - Pokud se jedná o server, musí mít definovaný status.

3.4.2 Příprava dat

```
ALTER TABLE `CmdbDevices`
      ADD COLUMN `Problem` VARCHAR(200) NULL AFTER `Status`;
```

Datová příprava pro toto upozornění bude spočívat v přidání sloupce Problem do tabulky CmdbDevices. Tato tabulka se synchronizuje jednou za 24 hodin. Pokaždé když se synchronizuje, zavolá se dotaz, který zaplní sloupec Problem.



Obrázek 16: Cmdb Problem - synchronizace

Select 1:

```
SELECT
   b.DeviceId,
   b.DeviceType,
   b.SnapshotDate,
   b.DeviceName,
   b.InstanceId,
   b.ClientTriname,
   concat_ws(', ',
      if(b.BadDeviceName =1, 'DeviceName',NULL),
      if(b.MissingOS =1, 'OS missing',NULL),
      if(b.MissingStatus =1, 'Status missing',NULL),
      if(b.MonitoredAdress =1, 'Monitored Adress has incorect format',NULL),
      if(b.MissingDomain =1, 'Domain missing ',NULL),
      if(b.DeviceNameDuplicite =1, 'DeviceNameDuplicite ',NULL)
   ) as 'Problem'
FROM
   (
      VNITŘNÍ SELECT
```

```
   ) AS b
```

Kde vnitřní select je:

```
SELECT
   *,
```

Kontrola jméno zařízení:

```
   CASE cmdb.`Connection`
      WHEN 'Yes' THEN
         !(cmdb.DeviceName REGEXP
           '^[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}$')
      ELSE
         (RIGHT(cmdb.DeviceName,4) <> CONCAT('.',LOWER(cmdb.ClientTriname)))
   END AS 'BadDeviceName',
```

Kontrola pro servery a virtuální servery, definovaný operační systém:

```
   CASE
      WHEN cmdb.DeviceType IN ('Device','Virtual Device') THEN
         ((TRIM(cmdb.OsType) = '') OR (cmdb.OsType is null))
      ELSE 0
   END AS 'MissingOS',
```

Kontrola operační systém – doménové jméno u serverů s Windows:

```
   (
      (cmdb.OsType = 'Windows') AND
      (cmdb.Notes NOT LIKE '%Domain%')
   ) AS 'MissingDomain'
```

Kontrola monitorovaná adresa:

```
   (!
      (cmdb.MonitoredAddress REGEXP
         '^[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}$')
   ) AS 'MonitoredAdress',
```

Kontrola Status – existence:

```
   CASE
      WHEN cmdb.DeviceType IN ('Device','Virtual Device') THEN
         ((TRIM(cmdb.`Status`) = '') OR (cmdb.`Status` IS NULL))
      ELSE 0
   END AS 'MissingOS',
```

Kontrola DeviceName - unikátkost:

```
   CASE device_name_bad.DeviceNameCount
      WHEN device_name_bad.DeviceNameCount IS NOT NULL THEN 1
      ELSE 0
   END AS 'DeviceNameDuplicite'
FROM
   IPadministration.CmdbDevices cmdb
   LEFT JOIN(
      SELECT
         cmdb.DeviceName as 'DeviceName3',
         1 as DeviceNameDuplicite,
         cmdb.SnapshotDate as 'SnapshotDate3',
         cmdb.ClientTriname as 'ClientTriname3',
         COUNT(*) as 'DeviceNameCount'
      FROM
         IPadministration.CmdbDevices cmdb
      WHERE
         cmdb.SnapshotDate = CURRENT_DATE
      GROUP BY
         cmdb.DeviceName
      HAVING COUNT(*) <> 1) device_name_bad
   ON cmdb.DeviceName = device_name_bad.DeviceName3
      AND cmdb.SnapshotDate = device_name_bad.SnapshotDate3
      AND cmdb.ClientTriname = device_name_bad.ClientTriname3
```

Update 1:

```
UPDATE
   IPadministration.CmdbDevices
SET
   Problem = SELECT1.Problem
WHERE
   SnapshotDate = CURRENT_DATE AND
   ClientTriname = SELECT1.ClientTriname AND
   DeviceId = SELECT1.DeviceId
```

3.4.3 Návrh interpretace ve webovém rozhraní

S připraveným sloupcem Problem v Cmdb katalogu bude sestavení upozornění vypadat následovně:

```
SELECT
   CASE
      WHEN COUNT(*) > 0 THEN
         CONCAT('
           There were found ',
           SUM(VNITRNI.Pocet), '
           issues in Cmdb catalogue on ',
           GROUP_CONCAT(VNITRNI.ClientTriname), '
           clients. Please solve them as fast as possible.'
         )
      WHEN COUNT(*) = 0 THEN
          'There were detected no issues in Cmdb catalogue.'
   END AS Message
FROM (
```

```
SELECT
   COUNT(*) as 'Pocet',
   Cmdb.ClientTriname
FROM
   IPadministration.CmdbDevices Cmdb
WHERE
   Cmdb.Problem <> '' AND
   Cmdb.SnapshotDate = CURRENT_DATE
GROUP BY
   Cmdb.ClientTriname)VNITRNI
```

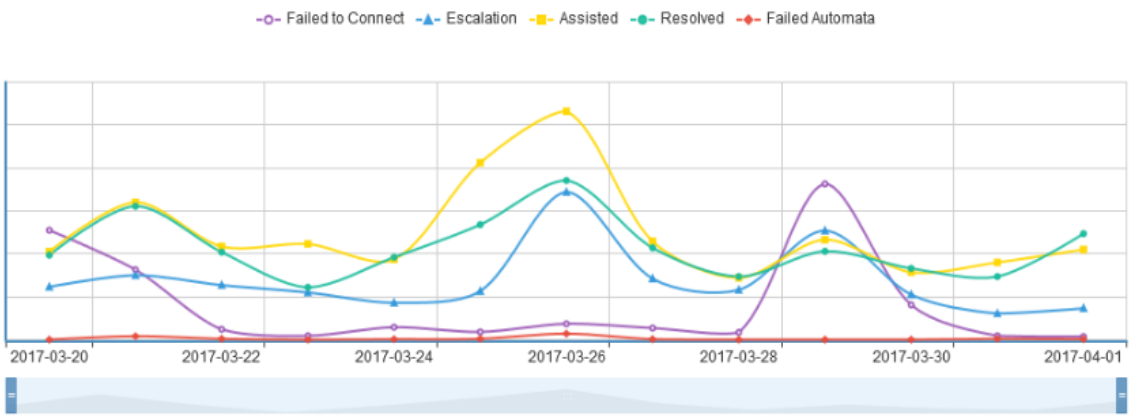
3.5 Prezentace finálního řešení

Finální řešení obsahuje tři druhy upozornění:

- navádění uživatele při hledání příčin odchylek,
- upozorňování na největší příležitosti pro řešení eskalovaných ticketů,
- upozorňování na problémy v CMDB katalogu.

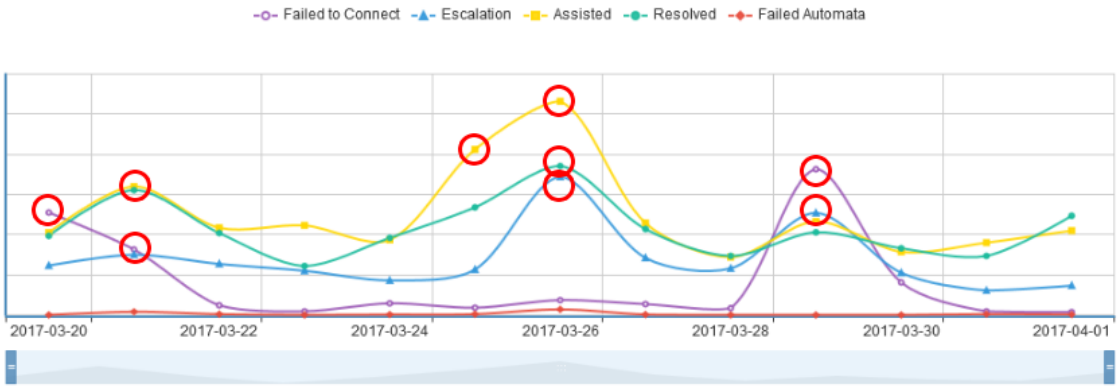
3.5.1 Odchylky

Graf exekucí pro náhodně vybranou skupinu:



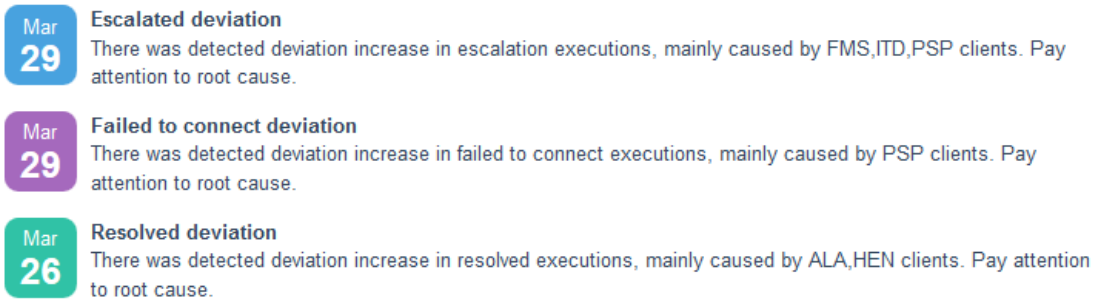
Obrázek 17: Graf exekucí

Detekované body:



Obrázek 18: Graf exekucí - detekované odchylka

Sestavené zprávy:

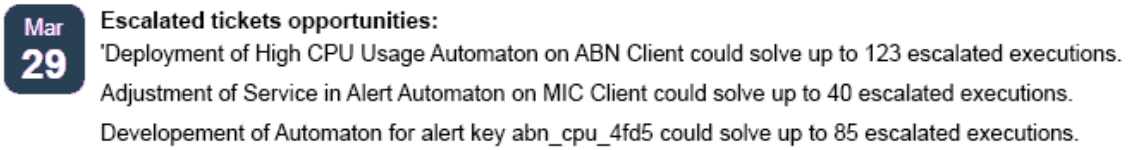


Obrázek 19: Upozornění – odchylky

Upozornění v sekci highlights navádí uživatele na jednotlivé klienty, které je doporučeno důkladněji analyzovat.

3.5.2 Příležitosti na eskalované tickety

Sestavené zprávy:



Obrázek 20: Upozornění – příležitosti

Uživatel tedy dostává informace o nejvýraznější alert key eskalovaných ticketů rozdělených do tří kategorií:

- největší příležitost pro vývoj nového automatu,

- největší příležitost pro nasazení nového automatu,
- největší příležitost pro upravení konfigurace dosavadního automat.

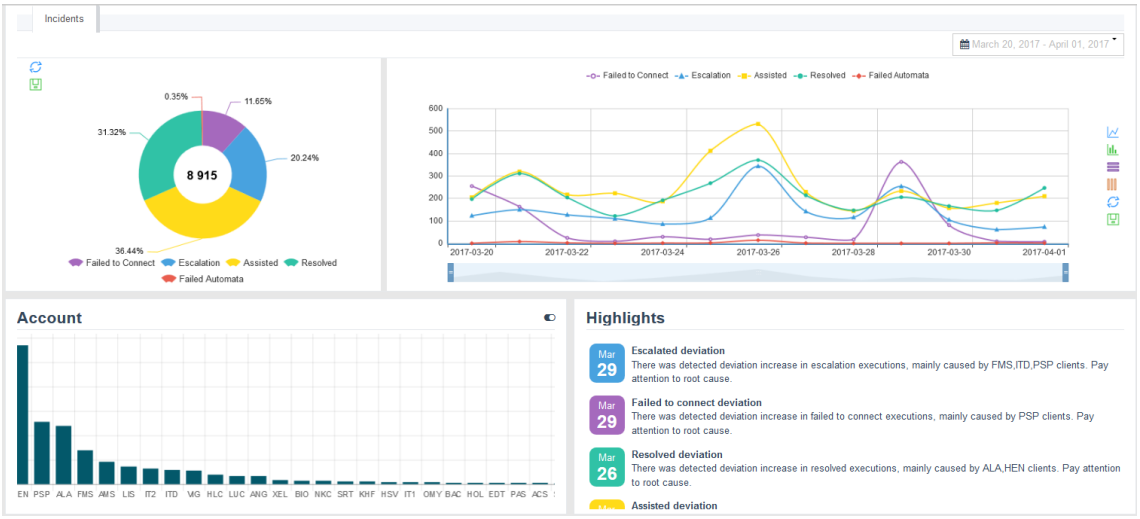
3.5.3 Problémy v CMDB katalogu

Obrázek 21: Cmdb problems

Uživatel je upozorňován na detekovatelné problémy v CMDB katalogu.

3.5.4 Nová verze dashboardu

Finální verze nového dashboardu bude vypadat následovně:



Obrázek 22: Dashboard 3.0

3.6 Ekonomické zhodnocení

V této části nejprve zhodnotím přínosy pro reportovací aplikaci a následně spočítám náklady, které musí společnost vynaložit na realizaci tohoto projektu.

3.6.1 Přínosy mé práce

Přínosy navrhovaného řešení:

- **Ušetření práce analytickému týmu**
    - Analýza odchylek analytickému týmu usnadňuje navigaci v programu, čímž zvyšuje efektivitu jeho práce. Analýza eskalovaných ticketů byla prováděna ručně a následně byla odesílána každý týden zákazníkům. Odstranění této rutinní činnosti dojde k časové úspoře analytického týmu.
- **Tlak na zákazníky řešit problémy**
    - Pomocí zobrazování problémů v Cmdb katalogu při přihlášení do aplikace, se vytvoří pasivní tlak na zákazníky, aby si vyřešili své problémy. Je možné očekávat zrychlení jejich práce, takže i zlepšení výsledků automatizace.
- **Kvalitní marketingové podklady**
    - Na budoucnost reportovacího nástroje je ve společnosti vyvíjen neustálý tlak, převážně díky existenci konkurenčních nástrojů. Produkt má teď nově argument, že už není pouze reportovacím nástrojem, ale že nově i navádí uživatele, jak některé problém řešit.
- **Potenciál pro další zlepšování**
    - Tato sekce představuje potenciál pro budoucí vývoj aplikace. Je možné do ní umístit témř jakoukoliv informaci, na kterou je nutné uživatele rychle upozornit.
- **Zlepšení výsledků automatizace**
    - Všechny výše zmíněné věci mají nepřímí vliv na zlepšení výsledků automatizace.

3.6.2 Náklady práce

Tabulka 1: Náklady na práci

Pracovník	**Práce**	**Hodin**	**Cena**
Čmok Petr	Počáteční analýza	40	16 000 Kč
Čmok Petr	Tvorba návrhu	20	8 000 Kč
Hlavní analytik	Konzultace	10	10 000 Kč
Čmok Petr	Úprava návrhu	5	2 000 Kč
Čmok Petr	Tvorba databázových struktur	5	2 000 Kč
Čmok Petr	Tvorba synchronizačních skriptů	40	16 000 Kč
Čmok Petr	Tvorba zadání pro front-end programátora	20	8 000 Kč
Front-end developer	Zpracování zadání	8	4 800 Kč
Čmok Petr	Zpětná analýza, prezentace výsledků	4	1 600 Kč

Odhadované náklady na projekt: **68 400 Kč.**

ZÁVĚR

V teoretickém úvodu práce byly představeny základní pojmy z oblastí informačních systémů, databázových systémů, automatizací a vybraných statistických metod. Analytická část práce dává tyto informace do praktického kontextu a stručně popisuje automatizační proces ve světově uznávané společnosti. Druhá část analýzy je věnována představení reportovacího nástroje, včetně jeho databázových struktur, o něž se opírá návrh vlastního řešení. Z analytické části vyplynulo několik problémů, jejichž řešení bylo představeno ve třetí části práce.

První problém s navigací na dashboardu v nové verzi aplikace byl vyřešen pomocí analýzy odchylky. Tento systém navádí uživatele do podskupin, které se nejvíce podílely na vzniku dané odchylky. Pokud analýza používání nové verze aplikace ukáže, že uživatelé tuto funkci využívají, bude možné ji rozpracovat do větší úrovně, kde uživatelé dostanou informace i o konkrétním automatu, alert key, nebo serveru, který odchylku způsobil.

Druhá část návrhu řešení představuje nejvýznamnější alert key z pohledu eskalovaných ticketů vybrané skupiny klientů. Tvoří pro ně tři druhy upozornění – nejvýznamnější příležitost z pohledu vývoje nového automatu, z pohledu nasazení nového automatu a z pohledu úpravy konfigurace stávajícího automatu. Toto řešení navádí uživatele, jak vylepšit výsledky analyzované skupiny.

Poslední řešený problém upozorňuje uživatele na detekovatelné chyby v CMDB katalogu. Vedení automatizačního týmu předpokládá, že se nasazením nové verze zrychlí korekce těchto chyb.

Práce obsahuje grafické návrhy, vývojové diagramy a příkazy v jazyce SQL, které budou sloužit jako podklady pro fornt-end programátory.

SEZNAM POUŽITÝCH ZDROJŮ

[1] HARDCASTLE, Elizabeth. *Business Information Systems* [online]. 2008 [cit. 2015-01-07]. ISBN 978-87-7681-463-2. Dostupné z: databáze bookboon.com

[2] CUESTA, H. a J. HUF. *Analýza dat v praxi*. 1. vyd. Brno: Computer Press, 2015, 296 s. ISBN 978-80-251-4361-2.

[3] WebFinance, Inc. What is information system? definition and meaning. *BUSINESS DICTIONARY* [online]. ©2015 [cit. 2015-01-07]. Dostupné z: http://www.businessdictionary.com/definition/information-system.html

[4] TechTarget. ICT (Information And Communications Technology - Or Technologies) Definition. *SearchCIE*. [online]. ©2007-2015 [cit. 2015-01-11]. Dostupné z: http://searchcio.techtarget.com/definition/ICT-information-andcommunications-technology-or-technologies

[5] BRUCKNER, Tomáš, Jiří VOŘÍŠEK, Alena BUCHALCEVOVÁ a kolektiv. *Tvorba informačních systémů*: *Principy, metodiky, architektury*. Praha: Grada Publishing, 2012. ISBN 978-80-247-4153-6.

[6] SODOMKA, Petr a Hana KLČOVÁ. *Informační systémy v podnikové praxi*. 2. vyd. Brno: Computer Press, 2010. ISBN 978-80-251-2878-7.

[7] BASL, Josef a Roman BLAŽÍČEK. *Podnikové informační systémy: Podnik v informační společnosti.* 2. vyd. Praha: Grada Publishing, 2008. ISBN 978-80-247-2279-5.

[8] Fakulta informatiky MU. *Životní cyklus informačního systému*. Fakulta informatiky Masarykovy Univerzity [online]. 2002 [cit. 2015-01-11]. Dostupné z: http://www.fi.muni.cz/~smid/mis-zivcyk.htm

[9] LACKO, Branislav, Ladislav MAIXNER, Pavel BENEŠ, Ladislav ŠMEJKAL. *Automatizace a automatizační technika: systemové pojetí automatizace* 1., vyd. Praha :Computer Press,2000. 1. vyd. 97 s. ISBN 80-7226-246-7.

[10] MERENDA, J. *Automatizace ve výrobních provozech.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2009. 43 s. Vedoucí bakalářské práce Ing. Marek Štroner, Ph.D.

[11] CORONEL, Carlos, Steven MORRIS a Peter ROB. *Database Systems: Design, Implementation and Management.* 9. vyd. Boston: Cengage Learning, 2009. ISBN 0538748842.

[12] HERNANDEZ, Michael J. *Návrh databází*. Praha: Grada, 2006. ISBN 80-247-0900-7.

[13] KROENKE, David, David J AUER a Jakub GONER. *Databáze*. Brno: Computer Press, 2015, ISBN 978-80-251-4352-0.

[14] Fakulta dopravní ČVUT v Praze. *Vztahy a relace, integrita*. ČVUT v Praze: Fakulta Dopravní [online]. 2007 [cit. 2015-02-19]. Dostupné z: https://it.fd.cvut.cz/k614daps/prednasky/dps_03_teorie.htm

[15] CONOLLY, Thomas, Carolyn E BEGG a Richard HOLOWCZAK. *Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází*. Brno: Computer Press, 2009. ISBN 978-80-251-2328-7.

[16] Microsoft. Základy normalizace databáze. *Microsoft: Support* [online]. ©2015 [cit. 2015-05-19]. Dostupné z: https://support.microsoft.com/en-us/kb/283878/cs

[17] PONNIAH, Paulraj. *Data warehousing fundamentals for IT professionals*. 2nd ed. Hoboken, N.J.: John Wiley, c2010. ISBN 978-0-470-46207-2.

[18] KIMBALL, Ralph a Margy ROSS. *The data warehouse toolkit: the definitive guide to dimensional modeling.* Third edition. ISBN 978-1-118-53080-1.

[19] MariaDB Knowledge Base. *Basic SQL Statements* [online]. 2016 [cit. 2017-05-07]. Dostupné z: https://mariadb.com/kb/en/mariadb/basic-sql-statements/

[20] HENDL, J. *Přehled statistických metod: analýza a metaanalýza dat*. 4. rozš. vyd. Praha: Portál, 2012. 734 s. ISBN 978-80-262-0200-4.

[21] NEUBAUER, Jiří, Marek SEDLAČÍK a Oldřich KŘÍŽ. *Základy statistiky: aplikace v technických a ekonomických oborech.* Praha: Grada, 2012, 236 s. : il. ; 24 cm. ISBN 978-80-247-4273-1.

[22] KROPÁČ, Jiří. *Statisitka C: statistická regulace, indexy způsobilosti, řízení zásob, statistické přejímky.* 2. přeprac. vyd. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2012, vi, 100 s. : il., grafy, tab. ISBN 978-80-7204-789-5.

[23] Dead database walking: MySQL's creator on why the future belongs to MariaDB. *Computer world* [online]. 2017: IDG Communications [cit. 2017-05-07]. Dostupné z: http://www.computerworld.com.au/article/457551/dead_database_walking_mysql_creator_why_future_belongs_mariadb.

SEZNAM TABULEK A OBRÁZKŮ

Seznam obrázků:

Obrázek 1: Vrstvy informačního systému, z [6] 15

Obrázek 2: Komponenty datového skladu, [17] 26

Obrázek 3: Správa serverů před automatizací 32

Obrázek 4: Správa serverů s automatizací 33

Obrázek 5: Automatizační proces 36

Obrázek 6: Dashboard 2.0 (testovací data) 42

Obrázek 7: Dashboard 3.0 skupina (testovací data) 43

Obrázek 8: Dashboard 3.0 – klient (testovací data) 44

Obrázek 9: 3.0 - ukázka reportu 45

Obrázek 10: Analýza odchylek - exekuce 47

Obrázek 11: Princip detekce odchylek 49

Obrázek 12: Synchronizační skript - DashboardValues 51

Obrázek 13: Dashboard - odchylky - webové rozhraní 53

Obrázek 14: AlertKeyOpportunity - synchronizace 59

Obrázek 15: AlertKey opportunity - webové rozhraní 61

Obrázek 16: Cmdb Problem - synchronizace 63

Obrázek 17: Graf exekucí 66

Obrázek 18: Graf exekucí - detekované odchylka 67

Obrázek 19: Upozornění – odchylky 67

Obrázek 20: Upozornění – příležitosti 67

Obrázek 21: Cmdb problems 68

Obrázek 22: Dashboard 3.0 68

Seznam tabulek:

Tabulka 1: Náklady na práci 70

SEZNAM PŘÍLOH

Příloha 1 – skript pro vytvoření výchozího prostředí

```
USE `IPadministration`;
CREATE TABLE `AccountOverview` (
	`Id` MEDIUMINT(9) NOT NULL AUTO_INCREMENT,
	`CDIR` VARCHAR(20) NULL DEFAULT NULL COMMENT 'Client Directory ID',
	`ClientId` SMALLINT(6) NOT NULL DEFAULT '0' COMMENT 'IPcenter Client Id',
	`ClientShort` VARCHAR(7) NOT NULL DEFAULT 'IBM' COMMENT 'Client Triname',
	`ClientName` VARCHAR(100) NOT NULL DEFAULT 'IBM' COMMENT 'Client Full
Name',
	`GroupId` SMALLINT(6) NOT NULL DEFAULT '0' COMMENT 'Group Id for Group',
	`GroupName` VARCHAR(50) NOT NULL DEFAULT 'IMT' COMMENT 'Common name for
IMT, Sector...',
	`GoLive` DATE NULL DEFAULT '2015-01-01' COMMENT 'Account Go Live Date',
	`AccountStatus` ENUM('Live','Sunset','Onboarding') NOT NULL DEFAULT
'Live' COMMENT 'Account Live status - Libe, Sunset, Onboarding...',
	`Devices` MEDIUMINT(9) NOT NULL DEFAULT '0' COMMENT 'Volume of Account
Active Devices',
	`CiWin` MEDIUMINT(9) NOT NULL DEFAULT '0' COMMENT 'Volume of Windows OS
Active Devices',
	`CiUnix` MEDIUMINT(9) NOT NULL DEFAULT '0' COMMENT 'Volume of Unix OS
Active Devices (Linix, HP-UX, Solaris, AIX...)',
	`CiOther` MEDIUMINT(9) NOT NULL DEFAULT '0' COMMENT 'Volume of Other OS
Active Device',
	`ConnRatio` DECIMAL(4,1) NOT NULL DEFAULT '0.0' COMMENT 'Connection Ratio
- output of CTK',
	`LastCTKCheck` VARCHAR(256) NOT NULL DEFAULT 'N/A' COMMENT 'Date and link
for last CTK test ',
	`LastCTKDate` DATE NULL DEFAULT NULL COMMENT 'Date for last CTK test',
	`NotCmdbR4E` ENUM('Escalate','Continue','N/A') NULL DEFAULT 'N/A' COMMENT
'Behaviour of Automaton than R4E has no recognised',
	`Snapshot` DATE NOT NULL DEFAULT '2015-01-01' COMMENT 'Snapshot date',
	`LastModification` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
	PRIMARY KEY (`Snapshot`, `ClientShort`),
	UNIQUE INDEX `Id` (`Id`),
	INDEX `CDIR` (`CDIR`),
	INDEX `ClientId` (`ClientId`),
	INDEX `ClientName` (`ClientName`),
	INDEX `GroupId` (`GroupId`),
	INDEX `GoLive` (`GoLive`),
	INDEX `AccountStatus` (`AccountStatus`),
	INDEX `LastModification` (`LastModification`),
	INDEX `NotCmdbR4E` (`NotCmdbR4E`),
	INDEX `GroupName` (`GroupName`),
	INDEX `LastCTKDate` (`LastCTKDate`)
)
COMMENT='Account Overview - daily snapshot for Auto Reports (Account Overview)'
COLLATE='utf8_general_ci'
ENGINE=MyISAM;

CREATE TABLE `AutomataExecutions` (
	`RadarTicket` INT(11) NULL DEFAULT NULL,
	`RadarStatus` VARCHAR(20) NOT NULL DEFAULT '',
	`CreatedDate` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
	`CloseDate` DATETIME NULL DEFAULT NULL,
	`StartDate` DATE NULL DEFAULT NULL COMMENT 'Date value of CreateDate',
	`ExecutionId` INT(11) NOT NULL DEFAULT '0',
	`InstanceId` TINYINT(4) NOT NULL DEFAULT '1',
	`AutomatonId` INT(11) NOT NULL DEFAULT '0',
	`Automaton` VARCHAR(255) NOT NULL DEFAULT '',
	`AutomataType` VARCHAR(50) NOT NULL DEFAULT '',
```

```
	`Description` VARCHAR(200) NULL DEFAULT NULL,
	`InstanceValue` VARCHAR(200) NULL DEFAULT NULL,
	`AlertKey` VARCHAR(100) NULL DEFAULT NULL,
	`AlertGroup` VARCHAR(50) NULL DEFAULT NULL,
	`EventType` MEDIUMINT(9) NULL DEFAULT NULL,
	`EventKey` VARCHAR(50) NULL DEFAULT NULL,
	`Summary` VARCHAR(512) NULL DEFAULT NULL,
	`TicketGroup` VARCHAR(50) NULL DEFAULT NULL,
	`ExecutionMode` VARCHAR(50) NOT NULL DEFAULT '',
	`TicketOutcome` VARCHAR(20) NOT NULL DEFAULT '',
	`ReturnCode` SMALLINT(6) NULL DEFAULT NULL,
	`ClosureCode` SMALLINT(6) NULL DEFAULT NULL,
	`FailedToConnect`	VARCHAR(1)	NOT	NULL	DEFAULT	''	COLLATE
'utf8mb4_general_ci',
	`OpportunityStatus` VARCHAR(25) NULL DEFAULT NULL,
	`ExecutionStatus` VARCHAR(25) NULL DEFAULT NULL,
	`Success` TINYINT(4) NOT NULL DEFAULT '0',
	`Status` VARCHAR(20) NOT NULL DEFAULT '',
	`StatusId` TINYINT(4) NOT NULL DEFAULT '0' COMMENT '1 = ABORTED, 2 =
ABORT_REQUESTED, 3 = AUTO_ABORTED, 4 =COMPLETE, 5 = FAILED, 6 = READY, 7 =
RUNNING, 8 = SCHEDULED',
	`IsCmdb`	ENUM('Y','N','N/A')	NOT	NULL	DEFAULT	'N/A'	COLLATE
'utf8mb4_general_ci',
	`QueueName` VARCHAR(20) NOT NULL DEFAULT '',
	`IpimTicket` INT(11) NOT NULL DEFAULT '0',
	`DiagnoseTime` VARCHAR(8) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
	`RemediateTime`	VARCHAR(8)	NULL	DEFAULT	NULL	COLLATE
'utf8mb4_general_ci',
	`eBondTicket` VARCHAR(20) NULL DEFAULT NULL,
	`ExternalTicketId` VARCHAR(40) NULL DEFAULT NULL,
	`IPC` VARCHAR(40) NULL DEFAULT NULL,
	`IMT` VARCHAR(10) NULL DEFAULT NULL,
	`ClientTriname` VARCHAR(5) NOT NULL DEFAULT '',
	`ClientName` VARCHAR(100) NOT NULL DEFAULT '',
	`ClientID` INT(11) NOT NULL DEFAULT '0',
	`Hostname` VARCHAR(100) NULL DEFAULT NULL,
	`LastModification` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
	PRIMARY KEY (`ExecutionId`, `InstanceId`, `ClientID`),
	INDEX `RadarTicket` (`RadarTicket`),
	INDEX `RadarStatus` (`RadarStatus`),
	INDEX `CreatedDate` (`CreatedDate`),
	INDEX `CloseDate` (`CloseDate`),
	INDEX `Automaton` (`Automaton`),
	INDEX `AutomataType` (`AutomataType`),
	INDEX `TicketOutcome` (`TicketOutcome`),
	INDEX `FailedToConnect` (`FailedToConnect`),
	INDEX `OpportunityStatus` (`OpportunityStatus`),
	INDEX `ExecutionStatus` (`ExecutionStatus`),
	INDEX `Status` (`Status`),
	INDEX `IsCmdb` (`IsCmdb`),
	INDEX `AlertKey` (`AlertKey`),
	INDEX `QueueName` (`QueueName`),
	INDEX `IpimTicket` (`IpimTicket`),
	INDEX `eBondTicket` (`eBondTicket`),
	INDEX `IMT` (`IMT`),
	INDEX `ClientTriname` (`ClientTriname`),
	INDEX `ClientName` (`ClientName`),
	INDEX `ClientID` (`ClientID`),
	INDEX `Hostname` (`Hostname`),
	INDEX `AlertGroup` (`AlertGroup`),
	INDEX `EventType` (`EventType`),
	INDEX `EventKey` (`EventKey`),
	INDEX `Success` (`Success`),
	INDEX `LastModification` (`LastModification`),
```

```
      INDEX `ReturnCode` (`ReturnCode`),
      INDEX `ClosureCode` (`ClosureCode`),
      INDEX `TicketGroup` (`TicketGroup`),
      INDEX `StatusId` (`StatusId`),
      INDEX `eBondTicketNetcool` (`ExternalTicketId`),
      INDEX `AutomatonId` (`AutomatonId`),
      INDEX `StartDate` (`StartDate`),
      INDEX `IPC` (`IPC`)
)
COMMENT='Agregated table (static view) - base table for reporting'
COLLATE='utf8_general_ci'
ENGINE=MyISAM
ROW_FORMAT=DYNAMIC;

CREATE TABLE IF NOT EXISTS `AutomataOverview` (
  `AutomatonID` int(11) NOT NULL,
  `InstanceId` tinyint(4) NOT NULL,
  `IMT` varchar(50) DEFAULT NULL,
  `Triname` varchar(5) NOT NULL,
  `AccName` varchar(100) DEFAULT NULL,
  `Automaton` varchar(100) DEFAULT NULL,
  `AutomatonClientCommonName` varchar(100) DEFAULT NULL,
  `Purpose` varchar(20) DEFAULT NULL,
  `AutomataShort` varchar(100) DEFAULT NULL,
  `Version` varchar(100) DEFAULT NULL,
  `AutomatonGoLive` date DEFAULT NULL,
  `AutomatonLastApproval` date DEFAULT NULL,
  `Matchers` varchar(15000) DEFAULT NULL,
  `30DaysExecs` mediumint(9) DEFAULT NULL,
  `TierProhibited` varchar(128) DEFAULT NULL,
  `LastModification` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  PRIMARY KEY (`AutomatonID`,`InstanceId`,`Triname`),
  KEY `Automaton` (`Automaton`),
  KEY `Purpose` (`Purpose`),
  KEY `30DaysExecs` (`30DaysExecs`),
  KEY `TierProhibited` (`TierProhibited`),
  KEY `Triname` (`Triname`),
  KEY `IMT` (`IMT`),
  KEY `AccName` (`AccName`),
  KEY `AutomataShort` (`AutomataShort`),
  KEY `LastModification` (`LastModification`),
  KEY `AutomatonGoLive` (`AutomatonGoLive`),
  KEY `AutomatonLastApproval` (`AutomatonLastApproval`),
  KEY `AutomatonClientCommonName` (`AutomatonClientCommonName`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='AutomataOverview';

CREATE TABLE `CmdbDevices` (
      `DeviceId` INT(11) NOT NULL DEFAULT '0' COMMENT 'CI_ID',
      `DeviceType` VARCHAR(15) NOT NULL DEFAULT 'Device' COMMENT 'CI_TYPE',
      `DeviceName` VARCHAR(100) NOT NULL DEFAULT 'Device' COMMENT 'NAME',
      `InstanceId` TINYINT(4) NOT NULL DEFAULT '1' COMMENT 'Instanceid',
      `ImtId` TINYINT(4) NULL DEFAULT NULL COMMENT 'Imt Id',
      `IMT` VARCHAR(50) NULL DEFAULT NULL COMMENT 'IMT',
      `ClientId` SMALLINT(6) NOT NULL DEFAULT '1' COMMENT 'ClientId',
      `ClientTriname` VARCHAR(3) NULL DEFAULT 'IBM' COMMENT 'TRINAME',
      `ClientName` VARCHAR(255) NULL DEFAULT 'IBM' COMMENT 'OWNING_CLIENT',
      `MonitoredAddress` VARCHAR(30) NULL DEFAULT '0.0.0.0' COMMENT
'MONITORED_ADDRESS',
      `Description` VARCHAR(256) NULL DEFAULT NULL COMMENT 'DESCRIPTION',
      `OsType` VARCHAR(20) NULL DEFAULT NULL COMMENT 'OS_TYPE',
      `Tier` VARCHAR(20) NULL DEFAULT NULL COMMENT 'TIER',
      `Notes` VARCHAR(256) NULL DEFAULT NULL COMMENT 'NOTES',
      `AssetId` VARCHAR(50) NULL DEFAULT NULL COMMENT 'ASSET_ID',
      `LocatedAtSite` VARCHAR(100) NULL DEFAULT NULL COMMENT 'LOCATED_AT_SITE',
```

```
	`IPCenterManaged` CHAR(5) NULL DEFAULT NULL COMMENT 'IPCENTER_MANAGED',
	`Monitored` CHAR(5) NULL DEFAULT NULL COMMENT 'MONITORED',
	`Status` VARCHAR(100) NULL DEFAULT NULL COMMENT 'STATUS',
	`SnapshotDate` DATE NOT NULL DEFAULT '2016-01-01' COMMENT 'SnapshotDate',
	`Updated` DATETIME NULL DEFAULT NULL COMMENT 'UPDATED',
	`LastModification` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP COMMENT 'Timestamp',
	PRIMARY KEY (`DeviceId`, `DeviceType`, `InstanceId`, `ClientId`,
`SnapshotDate`),
	INDEX `DeviceId` (`DeviceId`),
	INDEX `DeviceType` (`DeviceType`),
	INDEX `InstanceId` (`InstanceId`),
	INDEX `ClientId` (`ClientId`),
	INDEX `ClientTriname` (`ClientTriname`),
	INDEX `ClientName` (`ClientName`),
	INDEX `Status` (`Status`),
	INDEX `Updated` (`Updated`),
	INDEX `LastModification` (`LastModification`),
	INDEX `SnapshotDate` (`SnapshotDate`),
	INDEX `DeviceName` (`DeviceName`),
	INDEX `MonitoredAddress` (`MonitoredAddress`),
	INDEX `Description` (`Description`),
	INDEX `OsType` (`OsType`),
	INDEX `Tier` (`Tier`),
	INDEX `Notes` (`Notes`),
	INDEX `AssetId` (`AssetId`),
	INDEX `LocatedAtSite` (`LocatedAtSite`),
	INDEX `IPCenterManaged` (`IPCenterManaged`),
	INDEX `Monitored` (`Monitored`),
	INDEX `ImtId` (`ImtId`),
	INDEX `IMT` (`IMT`)
)
COMMENT='List of all devices (Devices, virtual devices) from CMDB'
COLLATE='utf8_general_ci'
ENGINE=MyISAM
ROW_FORMAT=DYNAMIC;

CREATE TABLE `Dashboard` (
	`ClientID` SMALLINT(5) UNSIGNED NOT NULL DEFAULT '0' COMMENT 'Client Id',
	`ClientTriname` CHAR(3) NOT NULL DEFAULT 'IBM' COMMENT 'Client Triname',
	`Date` DATE NOT NULL DEFAULT '1965-02-22' COMMENT 'Recorded Date',
	`AutomataRemediation` SMALLINT(6) NOT NULL DEFAULT '0' COMMENT 'Automata
- Remediation Purpose',
	`AutomataDiagnosis` SMALLINT(6) NOT NULL DEFAULT '0' COMMENT 'Automata -
Diagnosis Purpose',
	`ActiveCI` MEDIUMINT(9) NOT NULL DEFAULT '0' COMMENT 'Active CI based on
CMDB',
	`Connectivity` SMALLINT(6) NOT NULL DEFAULT '0' COMMENT 'Real connectivity
to the CI based on CTK',
	`IncidentResolved` MEDIUMINT(9) NOT NULL DEFAULT '0' COMMENT 'Resolved
Incidents',
	`IncidentAssisted` MEDIUMINT(9) NOT NULL DEFAULT '0' COMMENT 'Assisted
Incidents',
	`IncidentEscalation` MEDIUMINT(9) NOT NULL DEFAULT '0' COMMENT 'Escalated
Incidents',
	`IncidentFailedToConnect` MEDIUMINT(9) NOT NULL DEFAULT '0' COMMENT
'Failed To Connect Incidents',
	`IncidentFailed` MEDIUMINT(9) NOT NULL DEFAULT '0' COMMENT 'Failed
Incidents',
	`RadarTickets` MEDIUMINT(9) NOT NULL DEFAULT '0' COMMENT 'Radar Tickets
with Status = Complete, Failed',
	`LastCTKCheck` VARCHAR(256) NOT NULL DEFAULT 'N/A' COMMENT 'Date and link
for last CTK test',
	`LastCTKDate` DATE NULL DEFAULT NULL COMMENT 'Date for last CTK test',
```

```
	`LastModfication` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP COMMENT 'Last Modification',
	PRIMARY KEY (`ClientTriname`, `Date`),
	INDEX `ClientID` (`ClientID`),
	INDEX `ClientTriname` (`ClientTriname`),
	INDEX `Date` (`Date`),
	INDEX `LastModfication` (`LastModfication`)
)
COMMENT='Daily subsets of data for each Client and day for agregation on
Dashboard Page'
COLLATE='utf8_general_ci'
ENGINE=MyISAM;

CREATE TABLE `EventServers` (
	`IMT` VARCHAR(10) NOT NULL DEFAULT '0' COMMENT 'IMT',
	`ClientID` SMALLINT(5) UNSIGNED NOT NULL DEFAULT '0' COMMENT 'Client Id',
	`ClientTriname` CHAR(3) NOT NULL DEFAULT 'IBM' COMMENT 'Client Triname',
	`ClientName` VARCHAR(100) NOT NULL DEFAULT 'IBM' COMMENT 'Client Name',
	`Date` DATE NOT NULL DEFAULT '1965-02-22' COMMENT 'Recorded Date',
	`Hostname` VARCHAR(100) NOT NULL COMMENT 'Hostname or IP adrress',
	`Volume` MEDIUMINT(9) NOT NULL DEFAULT '0' COMMENT 'Volume of tickets per
server',
	`LastModfication` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP COMMENT 'Last Modification',
	PRIMARY KEY (`ClientTriname`, `Hostname`, `Date`),
	INDEX `ClientID` (`ClientID`),
	INDEX `ClientTriname` (`ClientTriname`),
	INDEX `Date` (`Date`),
	INDEX `LastModfication` (`LastModfication`),
	INDEX `ClientName` (`ClientName`),
	INDEX `IMT` (`IMT`),
	INDEX `Hostname` (`Hostname`),
	INDEX `Volume` (`Volume`)
)
COMMENT='Daily subsets of count of events for each Server'
COLLATE='utf8_general_ci'
ENGINE=MyISAM
ROW_FORMAT=FIXED;

CREATE TABLE `LiveAccounts` (
	`CDIR` VARCHAR(16) NOT NULL DEFAULT '0',
	`ImtID` TINYINT(4) NOT NULL DEFAULT '0' COMMENT 'IMT Id',
	`IMT` VARCHAR(50) NULL DEFAULT NULL COMMENT 'IMT name',
	`CountryID` SMALLINT(6) NOT NULL COMMENT 'Country Id',
	`Country` VARCHAR(100) NOT NULL COMMENT 'Country Name',
	`InstanceId` TINYINT(4) NOT NULL DEFAULT '1' COMMENT 'Instance Id',
	`ClientID` SMALLINT(6) NULL DEFAULT '0' COMMENT 'Client Id',
	`ClientTriname` VARCHAR(3) NOT NULL COMMENT 'Client Triname',
	`ClientName` VARCHAR(128) NOT NULL COMMENT 'Client Name',
	`GoLive` DATE NULL DEFAULT NULL COMMENT 'Go Live Date',
	`SunSet` DATE NULL DEFAULT NULL COMMENT 'Sunset Date',
	`AccountStatus` VARCHAR(15) NULL DEFAULT NULL COMMENT 'Account status
(Live, sunset, on board...)',
	`WikiUid` VARCHAR(50) NULL DEFAULT NULL,
	`LastModification` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
	PRIMARY KEY (`ClientTriname`),
	UNIQUE INDEX `ClientID` (`ClientID`),
	UNIQUE INDEX `WikiUid` (`WikiUid`),
	INDEX `ImtID` (`ImtID`),
	INDEX `IMT` (`IMT`),
	INDEX `CountryID` (`CountryID`),
	INDEX `Country` (`Country`),
	INDEX `ClientName` (`ClientName`),
	INDEX `GoLive` (`GoLive`),
```

```
	INDEX `LastModification` (`LastModification`),
	INDEX `InstanceId` (`InstanceId`),
	INDEX `SunSet` (`SunSet`),
	INDEX `CDIR_CD` (`CDIR`),
	INDEX `AccountStatus` (`AccountStatus`)
)
COMMENT='List of live Accounts (GoLive IS NOT NULL)'
COLLATE='utf8_general_ci'
ENGINE=MyISAM;


CREATE TABLE `LiveAutomas` (
	`ImtId` TINYINT(4) NOT NULL DEFAULT '0',
	`IMT` VARCHAR(50) NULL DEFAULT NULL,
	`InstanceId` BIGINT(20) NOT NULL DEFAULT '1',
	`ClientID` SMALLINT(6) NOT NULL,
	`ClientShort` VARCHAR(3) NOT NULL DEFAULT '',
	`ClientName` VARCHAR(128) NOT NULL DEFAULT '',
	`AutomatonID` BIGINT(20) NOT NULL DEFAULT '0',
	`Automaton` VARCHAR(255) NOT NULL DEFAULT '',
	`AutomatonClientCommonName` VARCHAR(255) NULL DEFAULT NULL,
	`Purpose` VARCHAR(11) NOT NULL DEFAULT '' COLLATE 'utf8mb4_general_ci',
	`AutomatonGoLive` DATE NULL DEFAULT NULL,
	`AutomatonLastApproval` DATE NULL DEFAULT NULL,
	`RulesMatch` VARCHAR(1) NULL DEFAULT NULL,
	`RuleDiffs` TEXT NULL,
	`Matchers` TEXT NULL,
	`AutoCommonName`	VARCHAR(100)	NULL	DEFAULT	NULL	COLLATE
'utf8_unicode_ci',
	`Category` VARCHAR(100) NULL DEFAULT NULL COLLATE 'utf8_unicode_ci',
	`ToBeDeployed` INT(11) NULL DEFAULT NULL,
	PRIMARY KEY (`InstanceId`, `ClientID`, `AutomatonID`),
	INDEX `ImtId` (`ImtId`),
	INDEX `IMT` (`IMT`),
	INDEX `ClientID` (`ClientID`),
	INDEX `ClientShort` (`ClientShort`),
	INDEX `Automaton` (`Automaton`),
	INDEX `Purpose` (`Purpose`),
	INDEX `RulesMatch` (`RulesMatch`),
	INDEX `AutoCommonName` (`AutoCommonName`),
	INDEX `Category` (`Category`),
	INDEX `ToBeDeployed` (`ToBeDeployed`),
	INDEX `AutomatonClientCommonName` (`AutomatonClientCommonName`),
	INDEX `AutomatonGoLive` (`AutomatonGoLive`),
	INDEX `AutomatonLastApproval` (`AutomatonLastApproval`)
)
COMMENT='Static table for LiveAutomata View'
COLLATE='utf8_general_ci'
ENGINE=MyISAM;
```